EXHIBIT 3
Filed Under Seal

CLEMENT SETH ROBERTS (STATE BAR NO. 209203)
croberts@orrick.com
BAS DE BLANK (STATE BAR NO. 191487)
basdeblank@orrick.com
ALYSSA CARIDIS (STATE BAR NO. 260103)
acaridis@orrick.com
EVAN D. BREWER (STATE BAR NO. 304411)
ebrewer@orrick.com
ORRICK, HERRINGTON & SUTCLIFFE LLP
The Orrick Building
405 Howard Street
San Francisco, CA 94105-2669
Telephone: +1 415 773 5700
Facsimile: +1 415 773 5759

SEAN M. SULLIVAN (*pro hac* vice)
sullivan@ls3ip.com
MICHAEL P. BOYEA (*pro hac* vice)
boyea@ls3ip.com
COLE B. RICHTER (*pro hac* vice)
richter@ls3ip.com
LEE SULLIVAN SHEA & SMITH LLP
656 W Randolph St., Floor 5W
Chicago, IL 60661
Telephone: +1 312 754 0002
Facsimile: +1 312 754 0003

Attorneys for Sonos, Inc.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

GOOGLE LLC,

Plaintiff and Counter-defendant,

v.

SONOS, INC.,

Defendant and Counter-claimant.

Case No. 3:20-cv-06754-WHA
Related to Case No. 3:21-cv-07559-WHA

REPLY EXPERT REPORT OF DOUGLAS C. SCHMIDT

VI. ASSERTED CLAIMS & CLAIM CONSTRUCTION

15. As discussed in my Opening Report, I understand Sonos is currently asserting that the following claims from the '033 Patent are infringed directly and indirectly by Google. These are the claims that I have been asked to opine on.

- "Computing device" claims: 1, 2, 4, 9, 11, 16
- "Computer-readable medium" claims: 12, 13

16. I previously provided my opinions regarding claim construction of the foregoing claims, including a summary of the Court's construction of the term "playback queue" and the Court's rejection of Google's proposed construction that the term "remote playback queue" is limited to a "third-party application."

17. Specifically, I understand that the claim term "playback queue" refers to a "list of multimedia content selected for playback," with the following characteristics:

- The playback queue is the list of media items that is used for playback;
- The playback queue contains the entire list of media items selected for playback;
- The playback queue is not being used merely to process the list of media items for playback; and
- The playback queue is the queue that "runs the show."

18. As I noted in my Opening Report, however, the '033 Patent's claims do not recite the term "multimedia content" like the '615 Patent's claims do. Instead, the '033 Patent's claims recite the term "media item." For purposes of the '033 Patent, therefore, I will interpret the Court's construction of "playback queue" (provided in the context of claim 13 of the '615 Patent) as "***a list of one or more media items selected for playback***." However, my opinions would remain the same under the Court's exact construction of "playback queue" provided in the context of claim 13 of the '615 Patent and applied verbatim by Dr. Bhattacharjee because a POSITA would understand that the term "multimedia content" is synonymous with the term "media item" in the context of the '033 and '615 Patents.

19. I also understand that Google sought leave from the Court to file supplemental claim construction briefing to enable Google to argue that the term "remote playback queue" means "a playback queue provided by a third-party application." However, I understand that the Court recently denied Google's request to file supplemental claim construction briefing. *See* Dkt. 432.

C. Dr. Bhattacharjee's Opinions Contradict Google & Dr. Bhattacharjee's Representations to the Court

36. Dr. Bhattacharjee expends a considerable amount of his effort in his overview explaining that (i) "manipulations" to the Sender's queue is "entirely local and are not based on inputs by a cloud server, and, indeed are not reflected back to any so-called queue on a YouTube server" (Bhatta. Rebuttal Report, ¶68; *see id.*, ¶69), (ii) media items (i.e., "Autoplay items" or "service-recommended videos") set for playback in non-casting/local playback mode may be different than those played after transitioning to casting/remote mode (*see id.*, ¶¶54, 56-57, 96), and (iii) "changes to the [Sender's] local playback queue are not reflected in any playlist." *Id.*, ¶70, *see also id.*, ¶¶61-64. Dr. Bhattacharjee's opinions are flawed for various reasons.

37. **First**, Dr. Bhattacharjee's opinions that "manipulations" to the Sender's queue is "entirely local and are not based on inputs by a cloud server, and, indeed are not reflected back to any so-called queue on a YouTube server" (Bhatta. Rebuttal Report, ¶68; *see id.*, ¶69) and "changes to the [Sender's] local playback queue are not reflected in any playlist" (*id.*, ¶70, *see also id.*, ¶¶61-64) are ultimately unpersuasive based on Google and Dr. Bhattacharjee's repeated representations to the Court.

38. As discussed before, Google and Dr. Bhattacharjee repeatedly represented to the Court that the accused YouTube systems ***use only*** a "remote playback queue" (also referred to by Google and Dr. Bhattacharjee as a "cloud queue"). Schmidt Rebuttal Report, ¶302. Google and Dr. Bhattacharjee also repeatedly represented to the Court that the existence of a "local playback queue" in a system is ***mutually exclusive*** of a "remote playback queue" (or "cloud queue") and *vice versa*. *Id.*, ¶303. Regardless, that a user can make changes at the Sender's queue is irrelevant because ultimately the list of media items selected for playback that is provided by the YouTube cloud infrastructure runs the show for the Sender and Receiver.

39. In fact, I understand that the Court accepted Google and Dr. Bhattacharjee's representations that a system cannot have both a "local playback queue" and a "remote playback queue"/"cloud queue" because "locally-stored information is merely a mirror reflecting a subset of what is happening in the cloud queue." Dkt. 316, 9-10. I understand that, according to the Court, a subset of media items (such as that stored by a playback device when used with the accused

YouTube apps) did not constitute a "playback queue" because they "merely provide the means to *process* the lists for playback. In short, "the cloud queue runs the show." *Id.*, 10.

40. **Second**, Dr. Bhattacharjee's assertions that media items (i.e., "Autoplay items" or "service-recommended videos") set for playback in non-casting/local playback mode may be different than those played after transitioning to casting/remote mode (Bhatta. Rebuttal Report, ¶¶54, 56-57, 96) are also irrelevant because the ***content*** of the "remote playback queue" is not determinative. *Infra* ¶¶170-74.

41. Dr. Bhattacharjee's overview is therefore flawed for these additional reasons.

D. Dr. Bhattacharjee's Characterization of Google's MDx Protocol Is Flawed

42. Dr. Bhattacharjee discusses Google's MDx protocol at paragraphs 75-79 of his Rebuttal Report. However, virtually all of his discussion is irrelevant to his opinions regarding alleged non-infringement. I also note that his discussion of the history of the MDx protocol is inaccurate and incomplete, including the dates that he claims certain events or milestones occurred. However, because his discussion has no bearing on his alleged non-infringement positions, I do not need to address it in this Report.

E. Dr. Bhattacharjee's Overview of YouTube on a User Device

43. Dr. Bhattacharjee's purports to provide an "overview" of the accused YouTube apps running on a user device. *See* Bhatta. Rebuttal Report, §IX.A. I note that Dr. Bhattacharjee's overview is incomplete and inaccurate.

1. Dr. Bhattacharjee's Overview of Playing Back Media on User Device

44. To start, Dr. Bhattacharjee's overview of playing back media on a user device via a YouTube app (Bhatta. Rebuttal Report, ¶¶45-70) is incomplete and inaccurate. In this regard, Dr. Bhattacharjee's "overview" focuses primarily on what he refers to as "User Playlists." Bhatta. Rebuttal Report, ¶45 ("Using the YouTube Main and YouTube Music application on their mobile devices, users may select media items (*e.g.*, videos or songs) and add them to a playlist. [Dr. Bhattacharjee] will refer to these playlists as 'UserPlaylists' or 'User-CreatedPlaylists.'), ¶46.

45. However, as I explained in my Opening Report, each YouTube app except for the YouTube TV app enables a user to select a collection of media items for playback at the Sender that the user does not create, such as service-provided playlists, albums, etc. *See, e.g.*, Schmidt Op.

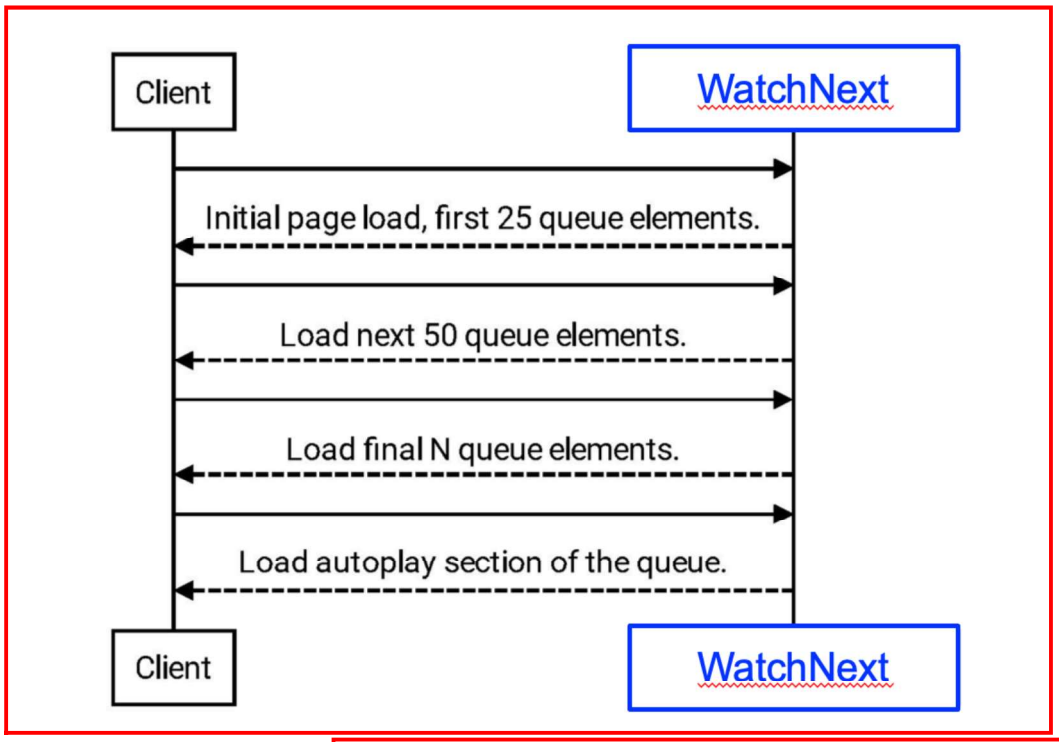
Report, ¶123. And of course, each YouTube app enables a user to select a single media item for playback at the Sender, and Google's auto-recommendations provide one or more next media items for the Sender to playback. *See, e.g.*, *id.*, ¶¶123-25.

46. Dr. Bhattacharjee's Rebuttal Report largely ignores these playback scenarios. It appears that Dr. Bhattacharjee does this to push the false narrative that a local playback queue runs the show for a YouTube Sender's playback. *See, e.g.*, Bhatta. Rebuttal Report, ¶¶60-70. But as I explained in my Opening Report, it is the YouTube cloud infrastructure –and specifically, the one or more Innertube servers that host the WatchNext, PlaylistDocumentService (PLDS), and PlaylistService services– that provides the list of media items that run the show for a YouTube Sender's playback (what I refer to as the "Watch Next queue"). *See, e.g.*, Schmidt Op. Report, ¶¶124-30, 152, 231-35, 248.

47. In this regard, in the non-Casting state, a Sender communicates with the WatchNext cloud service to obtain a first window of media items from the Watch Next queue that the Sender is set to playback.

48. In fact, Mr. Nicholson testified "I don't think its accurate" to say that the Sender's "Up Next" tab in YouTube Music is a representation of the "queue" because "[w]hen you have some renderers in your next up tab, it is almost certainly ***not an entire view into what the queue may be***. And you may have a 5,000 song playlist and you start playback, the next up tab is only going to contain -- I don't know the exact number -- but maybe 50, maybe 100 videos, songs, renderers." Nicholson Dep. Tr., 57:3-58:1. Mr. Nicholson went on to explain "[w]hat happens when you get to the bottom of that list [displayed in the 'Up Next' tab] is we get more RPCs to fetch ***more contents from YouTube servers to fill into that list***, and I model that as part of the users' listening experience." *Id; see also, e.g.*, *id.*, 40:21-41:4, 56:1-57:2.

49. As set forth in my Opening Report, Google's documents discuss and illustrate this process when a YouTube Sender initiates playback on a long playlist in which the Sender makes several calls to the WatchNext cloud server to obtain windows or "sections" of the Watch Next queue. *See* Schmidt Op. Report, ¶¶126-27.



GOOG-SONOSWDTX-00039785 [Server], 89 (annotated); Nicholson Dep. Tr., 58:9-25 (testifying that YouTube Music no longer uses a "GetMusicWatchNext" call and instead now uses a "GetWatchNext" call to the WatchNext service like YouTube Main), 71:3-73:12; *see also, e.g.*, GOOG-SONOSNDCA-00073352 [Queuing in YouTube Main App], 62 ("The look-ahead is ***extended continuously by watch next***. If the user keeps navigating forwards in the queue, they will be ***presented with an endless set of videos***, populated by the first item in watch next for the previous video.").

50. As I explained in my Opening Report, the WatchNext service provides data identifying next media items (videoIds) in the WatchNext queue by making calls (i.e., RPCs) to the PlaylistDocumentService that in turn makes calls to the PlaylistService. *See* Schmidt Op. Report, ¶¶126-28, 248. In this respect, the PlaylistService stores a list of media items selected for playback by the Sender in what Google refers to as a "BigTable," which is identified by a "playlistId," and the PlaylistService provides videoIds for media items from that stored list to the PlaylistDocumentService that returns the videoIds to the WatchNext service. *See, e.g., id.*; Nicholson Dep. Tr., 56:1-57:2:

> And that server makes RPCs into a service called the playlist service. And what we do is, ***we store a list of videos*** that the user is casting ***in a playlist in our playlist infrastructure***, which serves playlists across YouTube and those playlists ***have a playlist ID associated with them***. And all of these playlists when you're casting are

> prefixed with the letters RQ, and that stands for remote queue. So that is the -- a complete list of videos, a representation of these -- it is – it's on the server and what you see on the client is a representation of what is stored on YouTube servers.[4]

Nicholson Dep. Tr., 91:12-92:3:

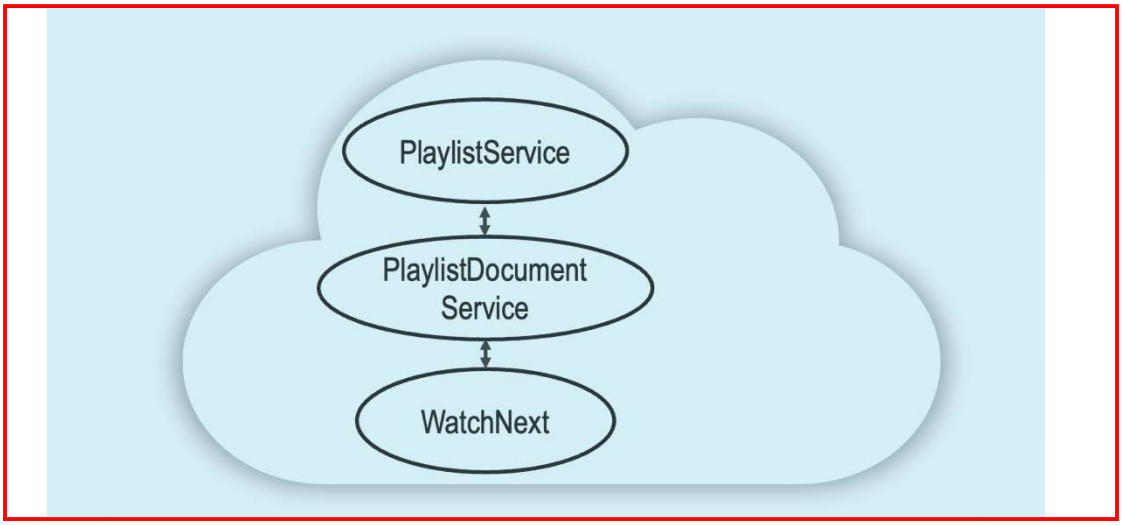
> **Q:** So where is the shared queue or remote queue stored?
> **A:** So the call number 2, the MDx session server makes a playlist modified call into the playlist service. What's not shown in this diagram are the implementation details of what happens within the playlist service. ***But the playlist service provides those list of video IDs into storage so that they can be later retrieved***.
> **Q:** What's the playlist service?
> **A:** So the playlist service is -- it's a service that exposes several APIs. One example is the playlist modify API allows users to modify a playlist that already exists. That's my understanding of the responsibilities in the playlist service.

See also, e.g., _rpc_manager.py // ln. 1735 (from InnerTubeWatchNext) ("Generate a request for playlist information from Playlist Document service."); playlist_document_service.proto // lns. 33-84 (handles a GetPlaylistDocumentRequest that can have a source set to "YTFE_WATCH_NEXT" indicating a request comes from a YT frontend WatchNext request for playback by a Sender); playlist_document_context.h // 41 ("Configures the request to playlist service and fetches the underlying YTPBPlaylist that the playlist document is going to be built out of."); playlist_service.h // lns. 78-79 ("The playlist stubby service implementation. A collection of RPCs for accessing and manipulating playlists stored in bigtable.").

51. For certain types of service-provided playlists, like radio- and mix-based playlists, the PlaylistService makes calls into another service for videoIds to add to the Watch Next queue. *See, e.g.*, GOOG-SONOSWDTX-00039785 [Server], 88; GOOG-SONOSWDTX-00039809 [WatchEndPoint]; Nicholson Dep. Tr., 49:10-50:19, 69:10-70:5, 73:13-74:9.

[4] While this testimony was in the context of the Sender Casting, the discussion of the PlaylistService storing the list of videos for playback is relevant to the non-Casting scenario as I explain below.

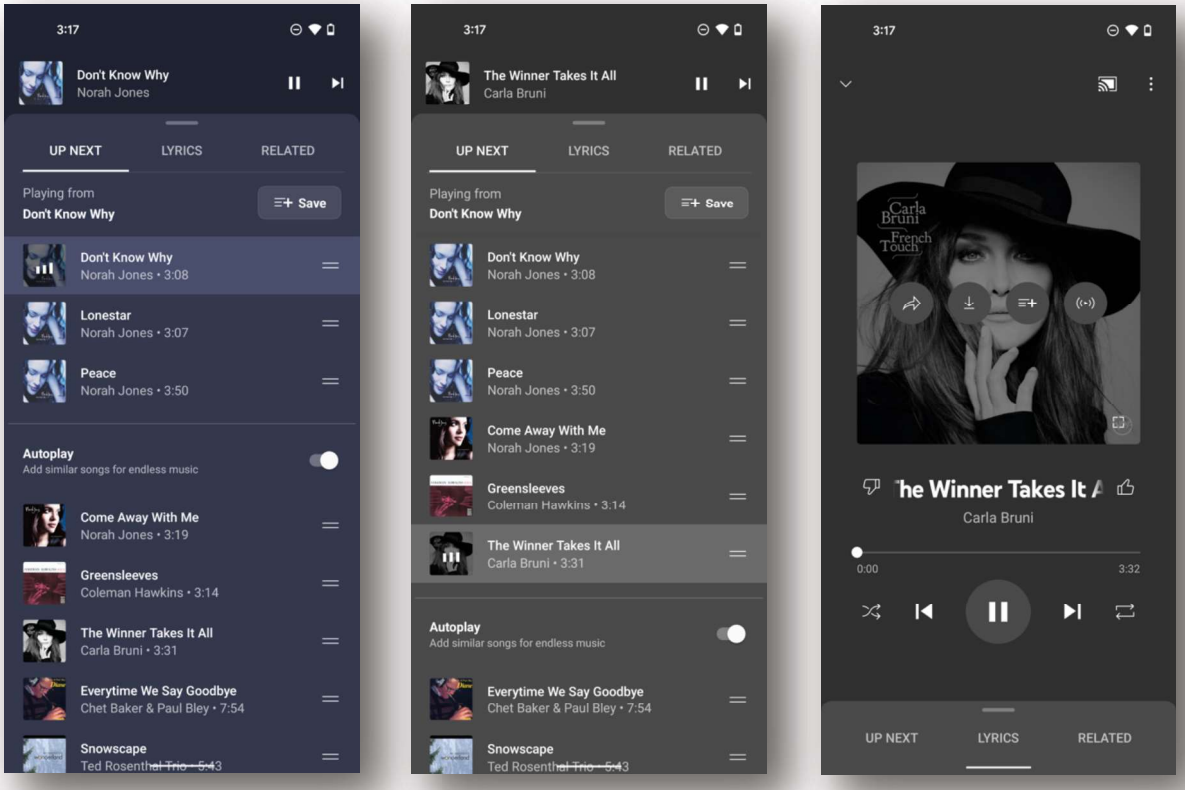
52. Google's YouTube Music Playback Squad documents titled "Server" and "The Queue" include some diagrams and flow diagrams illustrating the call flows between these services. *See* GOOG-SONOSWDTX-00039785 [Server], 88, 89, 90; GOOG-SONOSWDTX-00039798 [The Queue], 99. Mr. Nicholson explained that YouTube Music now just relies on the same WatchNext service as YouTube Main, so a more up-to-date, simplified diagram of the call flows between the WatchNext, PlaylistDocumentService, and PlaylistService services in the YouTube cloud infrastructure is as follows, with a Sender interfacing with the WatchNext service:



53. Unlike Google's prior art where a cloud server merely stored a catalog of "playlists," the YouTube cloud infrastructure provides a list of media items selected for playback by the Sender. In other words, unlike Google's prior art, while the list of media items is at the YouTube cloud infrastructure, it is designated for playback by the Sender. I will discuss some examples to highlight this point.

54. **First**, as I explained in my Opening Report, a user can select a service-provided playlist of videos using the YouTube Main app (as well as the YouTube Music and Kids apps) that gets added to the Watch Next queue for the Sender's playback. Schmidt Op. Report, ¶¶123, 126-130. As Mr. Nicholson explained, such a playlist could be thousands of media items long. *Supra* ¶48. The Sender receives, and displays a representation of, just a window of one or more videoIds from the Watch Next queue. This is demonstrated in the annotated screenshots shown below captured by a Pixel 5 device running the YouTube Main app where a service-provided playlist of videos was selected, and the Sender only displays the first 25 videos of that playlist.

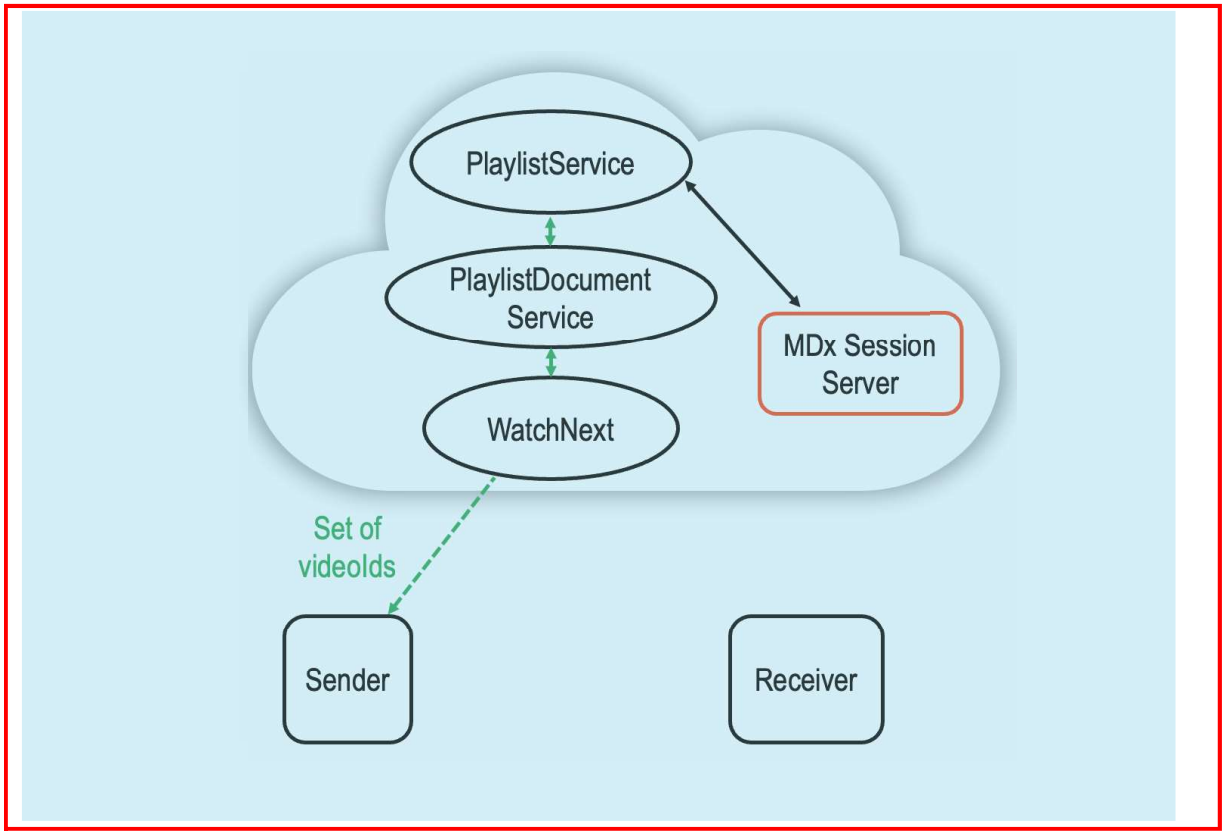
58. **Third**, as I explained in my Opening Report, a user can select a single media item or a collection of media items for playback at the Sender using the YouTube Music app and the YouTube cloud infrastructure will generate an "autoplay" list of media items selected for playback that are tailored for the user. Schmidt Op. Report, ¶125. The Sender receives, and displays a representation of, just a window of videoIds for this "autoplay" section of the Watch Next queue, as shown below on the left. Once the Sender plays through the media item(s) selected by the user (e.g., a single song, album, etc.), the Sender will exclusively play through the "autoplay" list of media items selected for playback by the YouTube cloud infrastructure, as shown below in the middle. As discussed in my Opening Report and in further detail below, the Sender can transfer playback of the "autoplay" list of media items selected for playback by the YouTube cloud infrastructure to a Receiver, as shown below on the right.



59. In this way, contrary to Dr. Bhattacharjee's opinions to the contrary, the local queue on the YouTube Sender ***merely provides the means by which the YouTube Sender processes*** the

Watch Next queue that is provided by the YouTube cloud infrastructure. The Watch Next queue runs the show for the YouTube Sender.

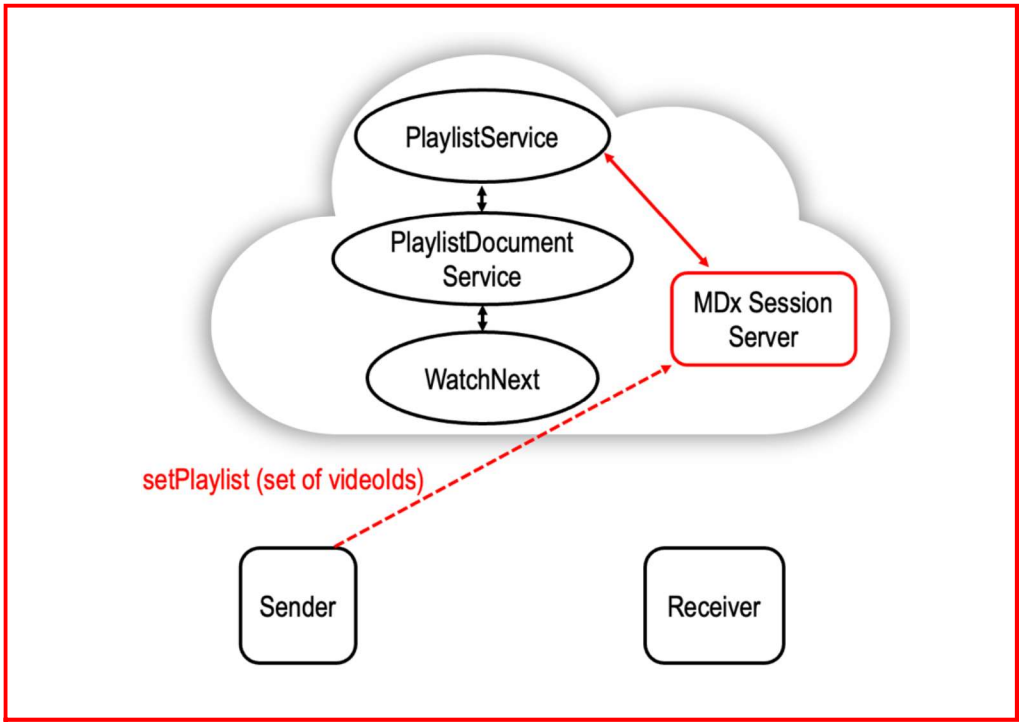
60. As I explained in my Opening Report, the fact that the Watch Next queue runs the show rather than the local queue on the YouTube Sender is demonstrated by the "client-side expansion" functionality that Google added[5] for Casting YouTube Music. *See* Schmidt Op. Report, ¶¶150-52. As I explained, "client-side expansion" generally first involves: (i) the YouTube Sender contacting the WatchNext service with a playlistId corresponding to the Watch Next queue provided by the YouTube cloud infrastructure, (ii) the WatchNext service making a call to the PlaylistDocument Service that in turn makes a call to the PlaylistService, which ultimately returns to the WatchNext service the set of videoIds for the media items in the Watch Next queue, and (iii) the YouTube Sender receiving the set of videoIds from the WatchNext service, as illustrated in the two below figures:



61. As I explained, "client-side expansion" next generally involves: (iv) the YouTube

[5] GOOG-SONOSWDTX-00039776 (announcement in June 2020 of clientside expansion on Android).

Sender sending that set of videoIds to the MDx session server as part of a setPlaylist message and (v) the MDx session server communicating with the Playlist Service to insert the set of videoIds into the Watch Next queue (or "RQ [remote queue] playlist"), as illustrated in the below figure:



62. Now one can readily appreciate that if, as Dr. Bhattacharjee contends, the local queue on the YouTube Sender ran the show, then "client-side expansion" would be completely ***unnecessary***, especially the YouTube Sender communicating with the WatchNext service (that in turn calls the PLDS and PlaylistService), because the YouTube Sender would already have the list of media items for playback locally at the YouTube Sender. In this way, "client-side expansion" demonstrates the existence of a list of media items selected for playback by the Sender that is provided by the YouTube cloud infrastructure that runs the show.

63. Likewise, as explained in my Opening Report, that the Watch Next queue runs the show rather than the local queue on the YouTube Sender is demonstrated by the fact that Casting YouTube Main does not involve "client-side expansion" or involve the Sender providing a list of media items for playback to the Receiver. *See* Schmidt Op. Report, ¶153. Instead, the Sender provides the MDx session server with a playlistId corresponding to the list of media items that was selected for playback by the Sender prior to the user initiating the Cast session, and the MDx session

server communicating with the Playlist Service to "expand" that playlistId into videoIds. *See, e.g.*, Schmidt Op. Report, ¶153; GOOG-SONOSWDTX-00039480 [Cast], 82.

2. Dr. Bhattacharjee's Overview of Casting to Receiver from User Device

64. Next, Dr. Bhattacharjee's overview of Casting to a Receiver from a user device via a YouTube app (Bhatta. Rebuttal Report, ¶¶71-90) is incomplete and inaccurate.

65. For instance, while Dr. Bhattacharjee discusses what Google refers to as the "Shared Queue"/"Remote Queue"/"MDx Queue" that the MDx session server manages once a Cast session is established (*see, e.g.*, Bhatta. Rebuttal Report, ¶80), Dr. Bhattacharjee glosses over details and explanation of what list of media items is driving the show for the Receiver's playback and the interplay between the MDx session server and other aspects of the YouTube cloud infrastructure

66. For example, as I noted before, Mr. Nicholson explained that the Shared Queue is actually stored via the Playlist Service.[6] *See, e.g.*, Nicholson Dep. Tr., 56:1-57:2:

> And that server makes RPCs into a service called the playlist service. And what we do is, ***we store a list of videos*** that the user is casting ***in a playlist in our playlist infrastructure***, which serves playlists across YouTube and those playlists ***have a playlist ID associated with them***. And all of these playlists when you're casting are prefixed with the letters RQ, and that stands for remote queue. So that is the -- a complete list of videos, a representation of these -- it is – it's on the server and what you see on the client is a representation of what is stored on YouTube servers.[7]

Nicholson Dep. Tr., 91:12-92:3:

> **Q:** So where is the shared queue or remote queue stored?
> **A:** So the call number 2, the MDx session server makes a playlist modified call into the playlist service. What's not shown in this diagram are the implementation details of what happens within the playlist service. ***But the playlist service provides those list of video IDs into storage so that they can be later retrieved***.
> **Q:** What's the playlist service?
> **A:** So the playlist service is -- it's a service that exposes several APIs. One example is the playlist modify API allows users to modify a playlist that already exists. That's my understanding of the responsibilities in the playlist service.

67. As another example, Google's documents explain that, while Casting, queue edit

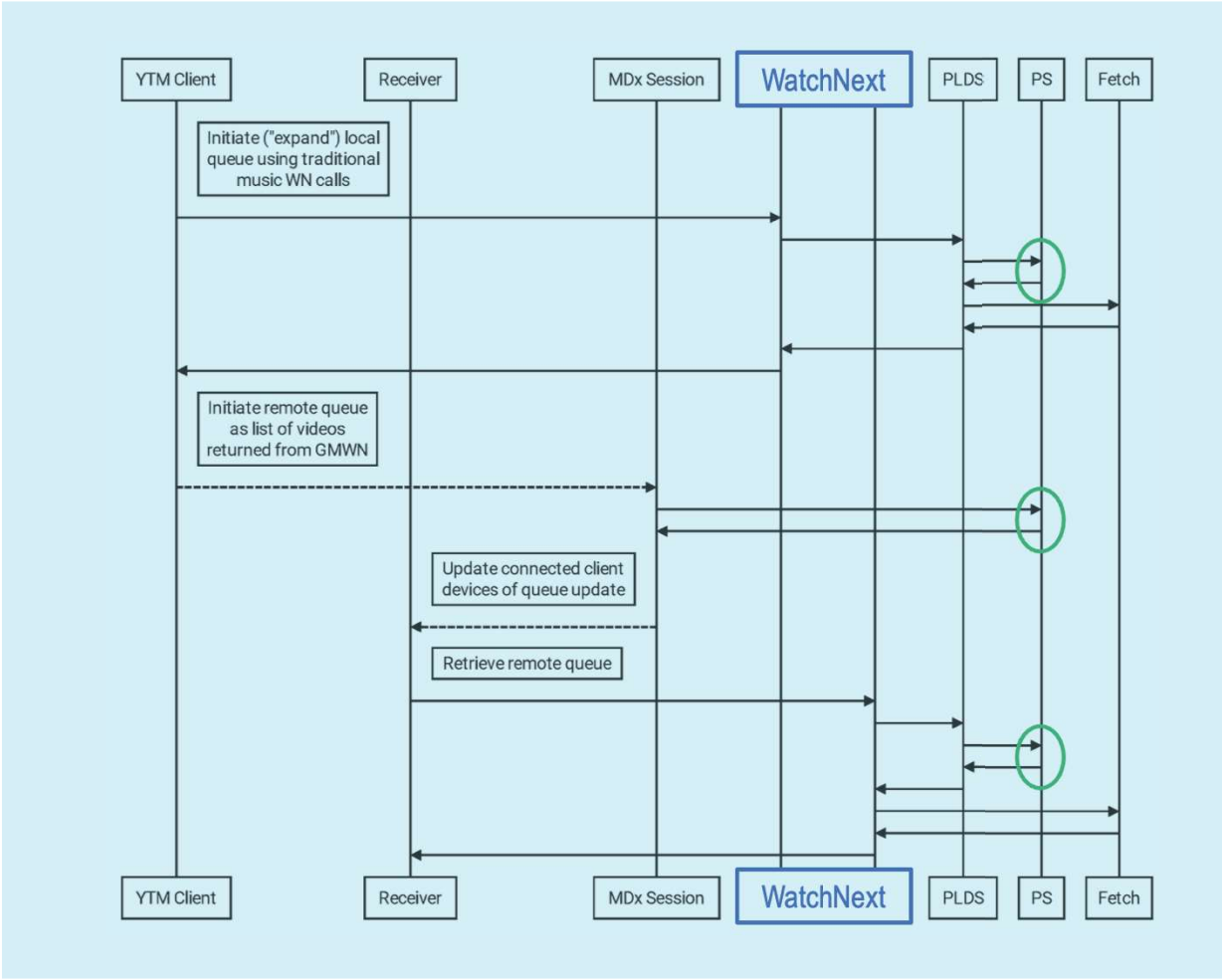
[6] I note that it appears that the MDx session server may have previously stored the Shared Queue locally but that responsibility has since been passed to the Playlist Service. *See, e.g.*, GOOG-SONOSWDTX-00039819 [Session Server]; GOOG-SONOSWDTX-00039542 [MDx-InnerTube Integration].

[7] While this testimony was in the context of the Sender Casting, the discussion of the PlaylistService storing the list of videos for playback is relevant to the non-Casting scenario as I explain below.

operations made by a user at a Sender result in the MDx session server making "calls to the PlaylistService to edit the underlying 'Remote Queue' playlist …." GOOG-SONOSWDTX-00039798 [The Queue], 800 ("The MDx Session Server manages the 'Remote Queue' playlist as well as the broader multi-device experience while Casting. Clients make queue edit operations (add to queue, re-order, remove from queue, etc) through the MDx Session Server, which makes appropriate calls to the PlaylistService to edit the underlying 'Remote Queue' playlist."); *see also, e.g.*, GOOG-SONOSWDTX-00041617 [YouTube Music Playback History in MDx Proposal], 20 ("The MDx Session Server adds the videoIds to the shared queue ('RQ' playlist) backed by the playlist service" and illustrating Session Server sending a "playlistModify" message to the Playlist Service), 24 ("MDx session server adds the videoEntries [(each containing a videoId)] to the shared queue ('RQ' playlist) via the Playlist Service" and illustrating Session Server sending a "playlistModify" message to the Playlist Service); GOOG-SONOSWDTX-00039827, 29, 32-33, 35

68. Moreover, if it was the case that the Shared Queue was resident on the MDx session server and that Shared Queue was running the show for the Receiver's playback, then the Receiver would not make WatchNext calls to the WatchNext service to obtain the next videoId. *See, e.g.*, GOOG-SONOSWDTX-00041617 [YouTube Music Playback History in MDx Proposal], 25 (illustrating and describing "WatchNext/Autoplay Flow for YTM Initiated MDx Playback" and explaining, when Receiver wants next video, "WatchNext Service requests the next video from the remote queue via the Playlist Document Service (PLSDS)."). But rather, the Receiver would obtain the next videoId from the MDx session server.

69. Thus, the evidence undermines Dr. Bhattacharjee's opinions that the YouTube cloud infrastructure does not provide a list of media items for playback by a Sender before Casting and for playback by a Receiver after Casting. This is summarized best by the following illustration showing the WatchNext, PLDS, and PlaylistServices providing a list of media items for playback before and after Casting:



GOOG-SONOSWDTX-00039480 [Cast], 82 (annotations added).

VIII. GOOGLE'S ACTS OF INFRINGEMENT

70. I previously summarized my opinions regarding Google's infringing conduct. *See* Schmidt Op. Report, ¶¶442-65. Dr. Bhattacharjee attempts to address these opinions in paragraphs 109-128 of his Rebuttal Report. However, Dr. Bhattacharjee's opinions are flawed for various reasons as discussed below.

A. Evidence of Google's Direct Infringement

71. Dr. Bhattacharjee's opinions regarding Google's acts of direct infringement are flawed for various reasons.

72. **First**, Dr. Bhattacharjee opines that "for purposes of direct infringement by Google ***only*** the Pixel Devices and the YouTube Main and YouTube Music applications are at issue" because I allegedly "only accuse[ ] Pixel Devices of direct infringement" at paragraph 442 of my Opening Report. Bhatta. Rebuttal Report, ¶109. This is a mischaracterization.

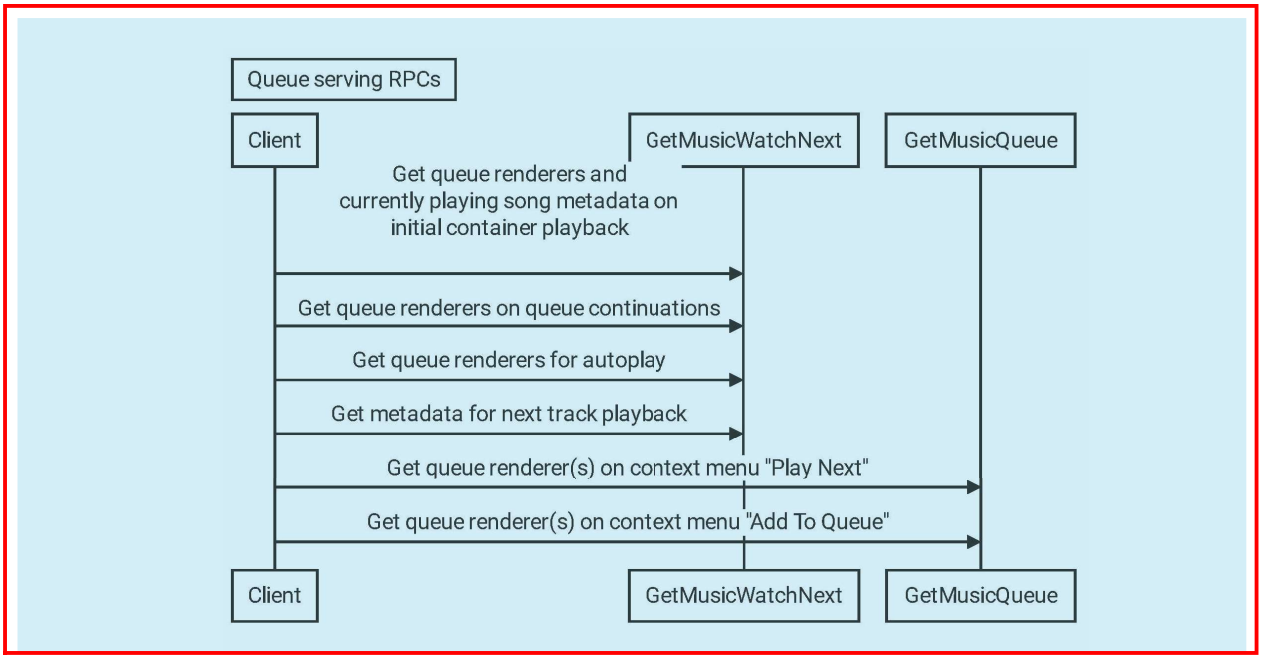
126. **First**, Dr. Bhattacharjee points to my prior opinions and evidence that reference a "Sender's local queue" (*id.*, ¶¶165-169) in concluding that "[a] User Device configured for playback of a local queue does not infringe." *Id.*, ¶164. However, Dr. Bhattacharjee's conclusion is based on the false premise that the existence of a "local queue" at the Sender precludes the existence of the claimed "remote playback queue." It does not.

127. As explained in my Opening Report, the Sender's local queue is ***loaded*** with data identifying one or more media items from the ***Watch Next queue***. *See, e.g.*, Schmidt Op. Report, ¶¶129 ("[T]he Sender's local queue is often ***loaded*** with (i) one or more 'videoIds' for the initial user-selected media item or collection of media items and (ii) one or more videoIds for service-recommended media item(s) seeded by the initial user selection."), 130 ("A representation of the Sender's local queue ***loaded*** with data identifying one or more media items from the ***Watch Next queue*** is typically displayed by the Sender.").

128. Indeed, the evidence I discussed in my Opening Report clearly shows that the Sender's local queue merely gets loaded with a ***window*** of media items from the Watch Next Queue (i.e., the list of media items selected for playback) provided by the WatchNext, PLDS, and Playlist Service, and it is the list of media items being provided by the WatchNext, PLDS, and Playlist Service that ***runs the show*** for the Sender' playback. *See, e.g.*, Schmidt Op. Report, ¶¶126-30; *see also supra* ¶¶46-52.

129. For example, contrary to Dr. Bhattacharjee's assertions at paragraph 166 of his report, "The Queue" document states that "[t]he YouTube Music queue is ***primarily*** a client-side construct" when not Casting, it does not say it is ***solely*** or ***exclusively*** a client-side construct. This makes perfect sense because, as with the other documents I cited, this document shows that the Sender only obtains ***windows*** of videoIds from the WatchNext service[9] at a time:

[9] Recall, Mr. Nicholson explained that the old GetMusicWatchNext was merged into the GetWatchNext used by YouTube Main.



GOOG-SONOSWDTX-00039798, 99.

130. As another example, contrary to Dr. Bhattacharjee's assertions at paragraph 166 of his report, the "YouTube Music Playback History in MDx Proposal" document (GOOG-SONOSWDTX-00041617) describes the "client-side expansion" process I explained before, which confirms that it is the list of media items selected for playback that is being provided by the WatchNext, PLDS, and Playlist Service that runs the show for the Sender's playback. *Supra* ¶¶46-52.

131. In other words, my opinions are consistent with the Court's order finding that "the cloud queue runs the show." Dkt. 316, 10.

132. Some of the documents cited by Dr. Bhattacharjee at paragraph 167 of his report also confirm that the Sender's local queue is loaded with data identifying one or more media items from the Watch Next queue, and thus, each Sender is "configured for playback of a remote playback queue," as recited in limitation 1.4. For instance, the "YTM Cast: Loop, aka Repeat" document confirms that the Sender's local queue contains only a window of media items from the Watch Next Queue provided by the WatchNext, PLDS, and Playlist Service and that the Sender uses subsequent WatchNext requests to obtain additional windows of media items (e.g., additional "autoplay" segments) from the Watch Next queue that runs the show. *See, e.g.*, GOOG-SONOSWDTX-00051490, 91 (comment [7]):

In short, when you initiate a new queue in music:

1. We make a WatchNext call with the video ID and/or playlist ID. The response contains the tabbed page structure and queue contents for the requested video/playlist.

2. A second WatchNext request is send by clients requesting the radio automix ('RDAMPL...' or 'RDAMVM...') for the previously requested container (or video if in the single song queue case). The response contains the autoplay preview contents that we tack on to bottom of the queue on client UIs.

So, eg when you start playback on an album and then open the queue and scroll to view your 5-song autoplay preview, you are seeing the results of two back-to-back WatchNext requests, one for the album, one for the RDAMPL mix of the album.

133. **Second**, Dr. Bhattacharjee opines that "in connection with the Patent Showdown on the '615 patent [I] repeatedly stated that a User Device plays back a 'local' queue in the non-Casting state." Bhatta. Rebuttal Report, ¶168; *see also id.*, ¶170. This is a mischaracterization.

134. Indeed, Dr. Bhattacharjee has not cited anything from my Prior Submissions where I opined that "a User Device plays back a 'local' queue in the non-Casting state." Instead, the opinions referenced by Dr. Bhattacharjee at paragraphs 168-169 of his report are consistent with my opinion that the Sender's local queue is loaded with data identifying one or more media items from the Watch Next queue, and thus, each Sender is "configured for playback of a remote playback queue," as recited in limitation 1.4.

135. Regardless, even assuming Dr. Bhattacharjee was correct that a Sender plays from a "local queue," it would merely reflect that the Sender's local queue ***provides the means*** for the Sender to ***process*** the Watch Next queue for playback, much in the same way the Receiver's storage of a window of the current, previous, and next videoIds for playback reflects the Receiver's ***means for processing*** the Watch Next queue rather than amounting to a "local playback queue," according to Dr. Bhattacharjee and the Court.

(3) "The Cloud Queue Runs the Show"

136. As discussed before, I understand that the Court accepted Google and Dr. Bhattacharjee's representations that a system cannot have both a "local playback queue" and a "remote playback queue"/"cloud queue" because "locally-stored information is merely a mirror reflecting a subset of what is happening in the cloud queue." Dkt. 316, 9-10. During the Patent Showdown summary judgment hearing, for example, I understand that Google specifically

represented to the Court that, in the accused products, "the phone is ***not*** involved in the processing of the queue or the maintenance of the queue … [T]he queue is up here in the Cloud." Transcript of Court Proceedings [Patent Showdown Summary Judgment Hearing] (July 13, 2022) ("Patent Showdown Hr'g Tr."), at 63-64. In short, as explained, the Court found that "the cloud queue runs the show" in the accused YouTube systems. *Id.*, 10.

137. However, Dr. Bhattacharjee now opines that the evidence "confirms that in the non-Casting use case a local queue, *not* a remote queue, is played back." Bhatta. Rebuttal Report, ¶170 (emphasis original). As discussed before, Dr. Bhattacharjee's opinion is simply not credible based on Google and Dr. Bhattacharjee's representations to the Court. *See* Schmidt Rebuttal Report, ¶¶302-303. Once again, the "local queue" on the Sender merely provides the means for the Sender to ***process*** the Watch Next queue for playback. The list of media items selected for playback that is provided by the WatchNext, PLDS, and Playlist Service runs the show for the Sender's playback.

138. Dr. Bhattacharjee opines "[t]hat the User Device is configured to playback the local queue—even where items loaded into that queue may have originated from a Watch Next message—is reflected by the fact that users can manipulate the queue without those changes being reflected back to the YouTube servers." Bhatta. Rebuttal Report, ¶171. I disagree. This, at best, shows that the means by which the Sender processes the Watch Next queue for playback (i.e., the Sender's "local queue") includes some additional functionality. But this does not detract from the fact that the list of media items selected for playback that is provided by the WatchNext, PLDS, and Playlist Service runs the show for the Sender's playback. Indeed, Dr. Bhattacharjee cannot dispute that the videoIds provided to the Sender by the WatchNext service dictates what the Sender plays back, regardless of whether a user manipulates a local copy of a window into the Watch Next queue.

139. Moreover, Dr. Bhattacharjee opines that I "appear[] to equate 'remote playback queue' with a storage area of a YouTube server that stores a playlist." Bhatta. Rebuttal Report, ¶172. This is a mischaracterization. In fact, I agree with Dr. Bhattacharjee that the mere storage of a "playlist" does not amount to a "playback queue." *See* Bhatta Rebuttal Report, ¶¶172-77.[10]

[10] As discussed above, however, Dr. Bhattacharjee's opinion that a "playlist" is not equivalent to a "playback queue" directly contradicts his invalidity opinions. *Supra* ¶¶32-34.

140. But the Watch Next queue provided by the YouTube cloud infrastructure is not merely a stored "playlist," as Dr. Bhattacharjee suggests. *Id.* Rather, while resident at the YouTube cloud infrastructure, the Watch Next queue comprises a list of media items selected for playback by the given Sender. In other words, while at the YouTube cloud infrastructure, this list of media items is designated for playback by the given Sender.

141. Indeed, the "autoplay" section of this list of media items is itself a list of media items specifically selected for playback by the user of the Sender based on various factors, such as the user's personal preferences or the like. *See, e.g.*, Nicholson Dep. Tr., 49:12-50:4, 50:11-19 ("The mix service is responsible ***for selecting*** which set of personalized songs to give to the user out of all of the videos that exist on YouTube."), 69:10-70:5 ("So the mixer service is responsible for generating the video IDs. It's Get Watch Next that's responsible for populating the renderers" at the Sender), 73:16-20 ("The mix service writes a list of videos into a – into storage."), 74:2-9; GOOG-SONOSNDCA-00073352 [Queuing in YouTube Main App], 56 ("Tap on a single video. An implicit list is created for the user, using autonav. We'll generate several look-ahead items to auto-extend the queue."), 62 ("The look-ahead is extended continuously by watch next. If the user keeps navigating forwards in the queue, they will be presented with an endless set of videos, populated by the first item in watch next for the previous video."); GOOG-SONOSWDTX-00005974 [Autoplay videos], 74 ("The Autoplay feature on YouTube makes it easier to decide what to watch next. After you watch a YouTube video, we'll automatically play another related video based on your viewing history."); GOOG-SONOSWDTX-00040744 [Watch Next + Autoplay], 44 ("**autoplay** (autoplaying the best watch next video, recommending more playlists") (emphasis original); GOOG-SONOSWDTX-00040628 [Product Excellence in Watch Next], 29; Schmidt Op. Report, ¶125. Dr. Bhattacharjee cannot dispute that a Sender is capable of playing back ***exclusively*** through a list of "autoplay" media items and that list is provided by the YouTube cloud infrastructure.

142. At bottom, while a YouTube Sender includes a "local queue," it is the list of media items selected for playback that is provided by the WatchNext, PLDS, and Playlist Service that ultimately runs the show.

- "**MR. VERHOEVEN [FOR GOOGLE]:** Yeah. And that just shows, ***[in the accused products] the phone is not involved in the processing of the queue or maintenance of the queue. The queue is maintained up here***, the phone gives the instruction, and the receiver calls for the first item in the queue. ***But the queue is up here in the Cloud.*** It used to be down here in the [playback] device, and for a variety of technical reasons, it moved to the Cloud, just like so many other things have moved to the Cloud. And so the queue used to be maintained in the remote device or the speaker playback device. Sonos did it that way. Google did it that way." Id., at 63-64.

- "Indeed, the accused applications are similar to the YouTube Remote and Tungsten/NexusQ prior art, with the exception that the prior art stored the playback queue locally on playback device (as required by claim 13) while ***the accused applications moved the playback queue to the cloud*** and thus do not infringe." Bhatta. Rebuttal Showdown Report, ¶387;

- "[A]s I discussed in my opening report, Google stored the playback queue locally on its receiver devices in its prior art products. ... However, in 2013 ***Google worked with Sonos to move the playback queue to the cloud***." *Id.*, ¶297;

Schmidt Rebuttal Report, ¶¶302, 307.

145. I therefore disagree with Dr. Bhattacharjee's attempt to limit Google and Dr. Bhattacharjee's representations to the Court as applying to only the casting mode and not the non-casting mode.

c. Dr. Bhattacharjee's Opinions Regarding a Hub Sender Are Flawed

(1) <u>Dr. Bhattacharjee Improperly Attempts to Limit My Infringement Opinions</u>

146. As an initial matter, Dr. Bhattacharjee incorrectly and improperly attempts to limit my infringement opinions of a Hub Sender to a scenario where "a Hub Device begin[s] playback when 'another Sender initiates a Cast session with the Hub Sender.'" Bhatta Rebuttal Report, ¶188. This is nonsense.

147. Nowhere in my Opening Report do I limit my opinions to a specific mechanism by which the Hub Sender begins playback. *See* Schmidt Op. Report, ¶178 (evidence indicating that Hub Sender can begin playing back media while operating in the local playback mode based on various triggers, including a voice input, another Sender initiating a Cast session, and user input at the Hub Sender's touchscreen display). Indeed, it would not even make sense in the context of claim 1 of the '033 Patent for me to have limited my opinions in this manner given that the claim

is directed to capability and limitation 1.4 merely requires a "computing device" be programmed with the functional capability to "operat[e] in a first mode in which the computing device is configured for playback of a remote playback queue provided by a cloud-based computing system associated with a cloud-based media service." In this regard, it does not require the "computing device" to be actively playing back from the "remote playback queue." Thus, it is no surprising that Dr. Bhattacharjee cites to no paragraph in my Opening Report to support his assertion. *See* Bhatta Rebuttal Report, ¶188.

148. Rather, I understand that it appears Dr. Bhattacharjee is taking this approach to attempt to support Google's damages expert's opinions that improperly remove all of "MDx voice" from the damages pool.

149. As another initial matter, Dr. Bhattacharjee opines "Dr. Schmidt does not discuss the playback path for a 'voice input' (or any other 'trigger'), and has not shown that a Hub Device is configured to play back a remote queue, as opposed to a local queue, when playback is initiated using a voice command." Bhatta. Rebuttal Report, ¶188. This is a remarkable opinion by Dr. Bhattacharjee given his and Google's repeated and vehement representations to the Court that Google's playback devices (that include Hub Devices) have ***no local playback queue at all*** in connection with any of the YouTube applications. *See also, e.g.*, Bhatta. Rebuttal Report, ¶¶172 ("[A]s [Dr. Bhattacharjee] showed in [his] declaration during the Patent Showdown, when Casting, a Cast-receiver may play back a cloud queue (also called a 'Shared Queue' or 'Remote Queue') implemented by the file SharedQueue.java. '615 Showdown Declaration, ¶¶50, 65, 73."), 185 ("[W]hen Casting the accused YouTube applications play back an 'MDx playback queue' (cloud queue) …."). Thus, Google's and Dr. Bhattacharjee's own admissions confirm that a Hub Device satisfies limitation 1.4.

(2) <u>Dr. Bhattacharjee Advances a Flawed, Brand New Non-Infringement Position</u>

150. Dr. Bhattacharjee opines "that a Hub Device is not a 'computing device' operating in the claimed 'first mode' when a User Device is Casting to the Hub Device" and "[t]he Hub Device is a Cast receiver device and is acting as a claimed 'playback device' in this case." Bhatta. Rebuttal Report, ¶¶187, 189-93. I disagree.

151. To start, I understand that this is a new non-infringement position that Google never raised during fact discovery. *Supra* ¶109. I note Google has no justification for not bringing this flawed non-infringement position before given that Google has been aware of Sonos's Hub Sender theory since at least October 2021. I offer my response here under the assumption that the Court does not strike Dr. Bhattacharjee's opinions on this topic.

152. **First**, Dr. Bhattacharjee assertion that my "opinion that a Hub Device is a 'computing device' operating in a 'first mode' during a Cast session is in tension with the other opinions in his report" (Bhatta. Rebuttal Report, ¶191) is misplaced. That the Hub Device has dual functionality where it can operate as a normal Receiver and also a Sender for other Receivers does not somehow mean it cannot satisfy the requirements of a "computing device" set forth by the '033 Patent. In this respect, for limitation 1.4, all that is required is the Hub Device be a "computing device" (which it indisputably is) with the programmed capability to "operat[e] in a first mode in which the [Hub Device] is configured for playback of a remote playback queue provided by a cloud-based computing system associated with a cloud-based media service" (which it indisputably has). *See, e.g.*, Bhatta. Rebuttal Report, ¶189 ("I agree that a Hub Device is playing back a cloud queue when in a Cast session with a mobile device ….").

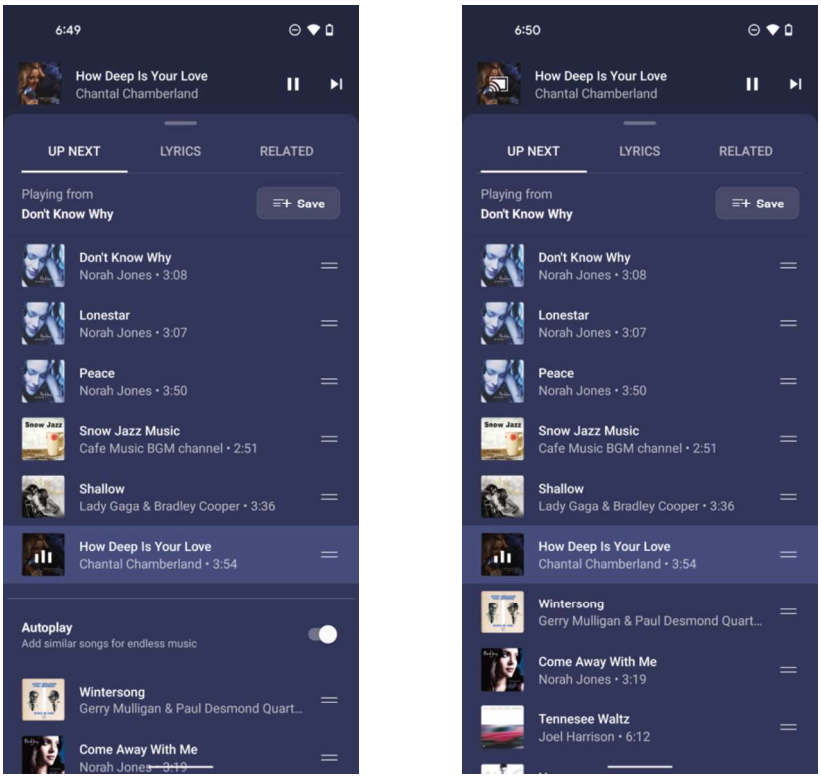
153. **Second**, Dr. Bhattacharjee has to distort the claim language to justify his flawed position. Bhatta. Rebuttal Report, ¶192. In this regard, after improperly asserting that my infringement analysis is limited to another Sender starting playback at the Hub Device, Dr. Bhattacharjee contends, "in this use case a Hub Device is a Cast receiver that plays back media when a computing device (e.g., a mobile device) transfers playback responsibility to the Hub Device (a playback device) and ***the system*** has transitioned out of the 'first mode.'" But the claims say nothing about the "***system***" operating in a "first mode." Rather, as I just explained, limitation 1.4 requires a "***computing device***" to have programmed capability to "operat[e] in a first mode in which the computing device is configured for playback of a remote playback queue provided by a cloud-based computing system associated with a cloud-based media service." The claim does not limit how the "computing device" must get into the "first mode." Thus, Dr. Bhattacharjee's opinions are baseless.

154. **Third**, Dr. Bhattacharjee incorrectly contends that his "opinion is supported by the specification of the '033 patent." Bhatta. Rebuttal Report, ¶193. To start, I understand that just because Google decided to put dual functionality into its Hub Devices that make such devices "playback devices" in the context of the '033 Patent generally and also infringe all of the limitations of claim 1 does not require the '033 Patent specification to have exact disclosure of such a device.

155. Moreover, Dr. Bhattacharjee ignores disclosures from the ''033 Patent that undermines his opinion. For example, the '033 Patent explains "[i]t should be noted that ***other*** networked-enabled devices such as [a] … smart phone ***or network-enabled device*** (***e.g.***, a networked computer such as a PC or Mac) can also be used as a controller to interact [*sic*] or control zone players in a particular environment." '033 Patent, 9:65-10:2. This disclosure informs a POSITA that the inventors contemplated a broad set of types of networked-computing devices that could control "zone players"/"playback devices." As another example, the '033 Patent explains "a zoneplayer can be ***part of another device***, which ***might even serve a purpose different*** than audio (***e.g.***, a lamp)." *Id.*, 9:11-13. Thus, I disagree with Dr. Bhattacharjee's assertion that the specification of the '033 Patent supports his opinions.

3. Each YouTube Sender Is Programmed with the Capability to [1.5] While Operating in the First Mode, Display a Representation of One or More Receivers in a Media Playback System & [1.6] While Displaying the Representation of the One or More Receivers, Receive User Input Indicating a Selection of at Least One Given Receiver

156. I explained in my Opening Report that each YouTube Sender (user device or Hub device provisioned with one or more YouTube apps) satisfies limitations 1.5 and 1.6. Schmidt Op. Report, ¶¶260-286. Dr. Bhattacharjee opines that I "failed to show that a User Device or Hub Device satisfies Limitations 1.5 and 1.6" because I have allegedly "failed to show that the accused YouTube applications can operate in a first mode in which the computing device is configured for playback of a remote playback queue provided by a cloud-based computing system associated with a cloud-based media service," as recited in limitation 1.4. Bhatta. Rebuttal Report, ¶194. Other than his opinions in connection with limitation 1.4, Dr. Bhattacharjee does **not** dispute any of my opinions that (i) a user device provisioned with any of the YouTube apps, and (ii) a Hub device provisioned with the YouTube Music app satisfies limitations 1.5-1.6. *See id.*, ¶¶194-196. As



175. Dr. Bhattacharjee also takes issue with the fact that the MDx session server manages a copy of the Watch Next queue apparently because "a 'copy of the Watch Next queue' is necessarily different than the original 'Watch Next queue.'" Bhatta. Rebuttal Report, ¶204. In this respect, Dr. Bhattacharjee appears to be arguing that the claimed "remote playback queue" cannot move, or in other words, where the "remote playback queue" is maintained in data storage in the "cloud-based computing system" must be fixed. I disagree with this, yet additional, new non-infringement position.

176. The claims require that the "remote playback queue" is provided by the same "cloud-based computing system associated with the cloud-based media service" before and after the transfer, but they do not otherwise say that the "remote playback queue" must be fixed at one data storage location within that "cloud-based computing system" or that it cannot otherwise move.

177. In the accused system, before Casting, the list of media items selected for playback by the Sender that runs the show is provided by the YouTube cloud infrastructure and specifically, by the WatchNext, PLDS, and Playlist Service with storage via the Playlist Service. *Supra* ¶¶46-52. Likewise, after Casting, the list of media items selected for playback by the Receiver that runs

the show is also provided by the YouTube cloud infrastructure and also specifically, by the WatchNext, PLDS, and Playlist Service with storage via the Playlist Service, although the MDx session server facilitates management of the list. *Supra* ¶¶60-63. In both cases, the WatchNext Service provides one or more videoIds of media items that the Sender or Receiver is to playback next, which are from the list of one or more media items selected for playback that is stored in the YouTube cloud infrastructure. Likewise, in both cases, the next videoIds from the list stored in the YouTube cloud infrastructure ultimately controls the Sender's or Receiver's playback.

178. Thus, there is nothing in Dr. Bhattacharjee's Rebuttal Report that changes my opinion that a user device provisioned with a YouTube app literally satisfies limitation 1.7 (and limitation 1.4).

179. However, even assuming *arguendo* that some or all of Dr. Bhattacharjee's new claim construction requirements and/or non-infringement positions were correct (they are not), it is my opinion that a user device provisioned with a YouTube app would still satisfy limitation 1.7 (and limitation 1.4) under the Doctrine of Equivalents.

180. **First**, this is because, in my opinion, there is an insubstantial difference between (i) the ***contents*** of a list of one or more media items selected for playback being the same before and after transfer and (ii) the ***contents*** of a list of one or more media items selected for playback being different after transfer as compared to before transfer.

181. In fact, the Sender performs the same function (e.g., operating in a first mode in which it is configured for playback of a list of one or more media items selected for playback provided by the YouTube cloud infrastructure), in the same way (e.g., by interacting with the YouTube cloud infrastructure providing the list of one or more media items selected for playback), to achieve the same result (e.g., playing back media items from the list of one or more media items selected for playback provided by the YouTube cloud infrastructure) irrespective of whether the ***contents*** of the list provided by the YouTube cloud infrastructure is the same or different before and after Casting.

182. Likewise, the Receiver performs the same function (e.g., obtaining from the WatchNext service of the YouTube cloud infrastructure a videoId of a next media item in the list

of one or more media items selected for playback provided by the YouTube cloud infrastructure), in the same way (e.g., by communicating with the WatchNext service of the YouTube cloud infrastructure), to achieve the same result (e.g., using the obtained videoId of the next media item for use in retrieving the next media item for playback) irrespective of whether the ***contents*** of the list provided by the YouTube cloud infrastructure is the same or different before and after Casting.

183. Additionally, the YouTube cloud infrastructure performs the same function (e.g., maintaining the list of one or more media items selected for playback by a Sender and Receiver), in the same way (e.g., by storing a list of one or more videoIds in data storage in the cloud), to achieve the same result (e.g., providing the list of one or more media items selected for playback by a Sender and Receiver, such as by providing the Receiver with a videoId identifying a next media item for playback in the list) irrespective of whether the ***contents*** of the list provided by the YouTube cloud infrastructure is the same or different before and after Casting.

184. **Second**, there is infringement under at least DoE because, in my opinion, there is an insubstantial difference between (i) the list of one or more media items selected for playback being stored in the same data storage location in "the cloud-based computing system" before and after transfer and (ii) the list of one or more media items selected for playback being stored in a different data storage location in "the cloud-based computing system" after transfer as compared to before transfer. Indeed, in both instances, the list of one or more media items selected for playback that is provided by "the cloud-based computing system" runs the show.

185. Moreover, the Sender performs the same function (e.g., operating in a first mode in which it is configured for playback of a list of one or more media items selected for playback provided by the YouTube cloud infrastructure), in the same way (e.g., by interacting with the YouTube cloud infrastructure providing the list of one or more media items selected for playback), to achieve the same result (e.g., playing back media items from the list of one or more media items selected for playback provided by the YouTube cloud infrastructure) irrespective of whether the list of one or more media items selected for playback is stored in the same or a different location in the YouTube cloud infrastructure before and after Casting.

186. Likewise, the Receiver performs the same function (e.g., obtaining from the

Dated: January 23, 2023

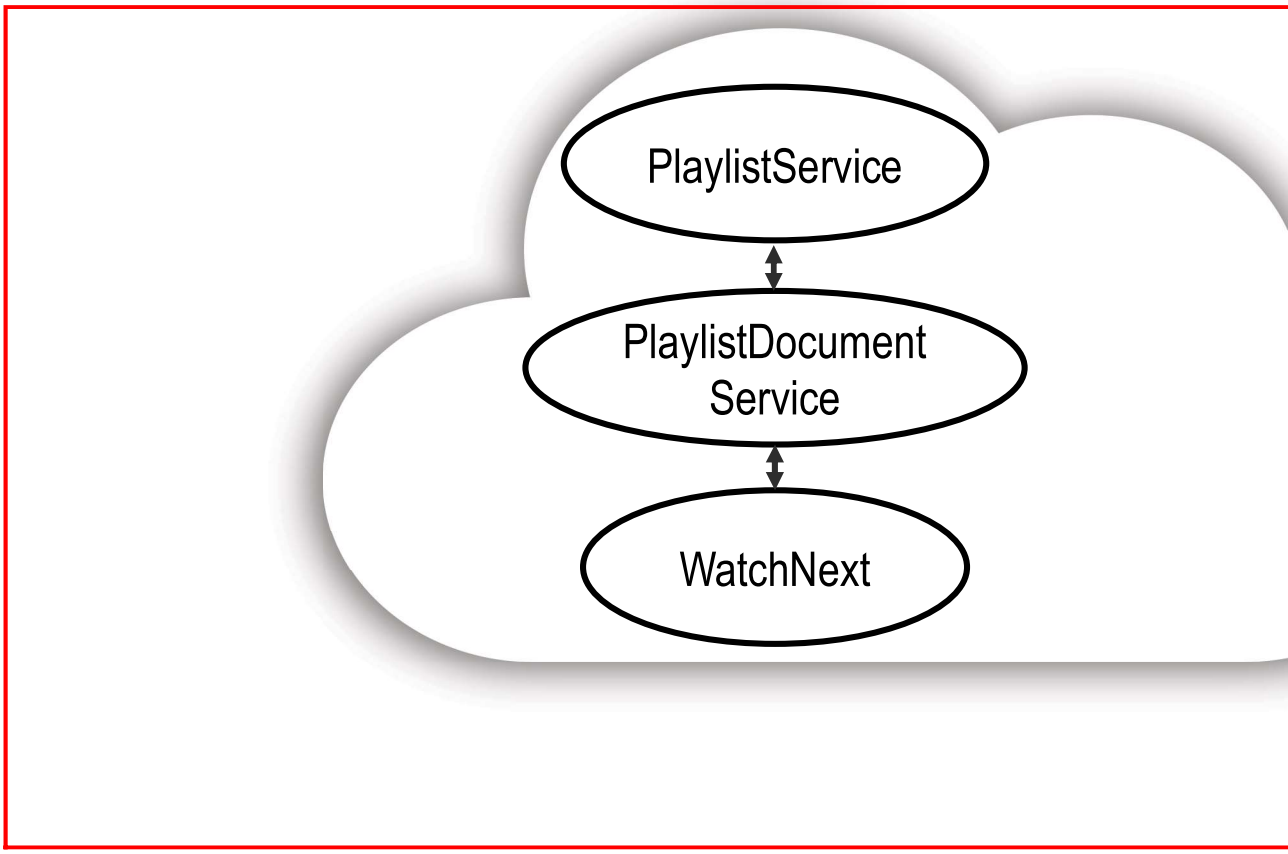
DOUGLAS C. SCHMIDT

Appendix 4

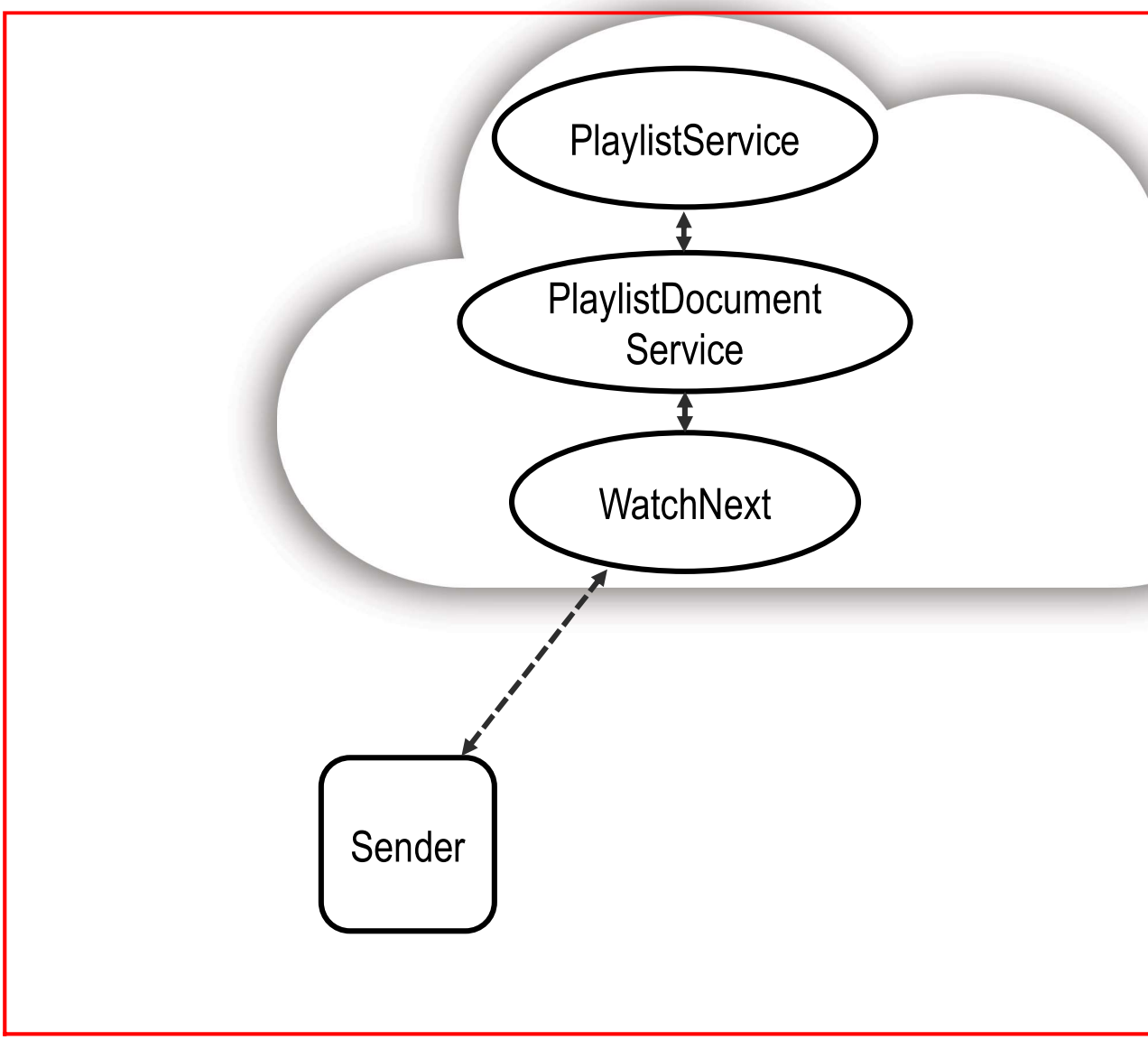
Expert Testimony of Dr. Douglas C

SONOS

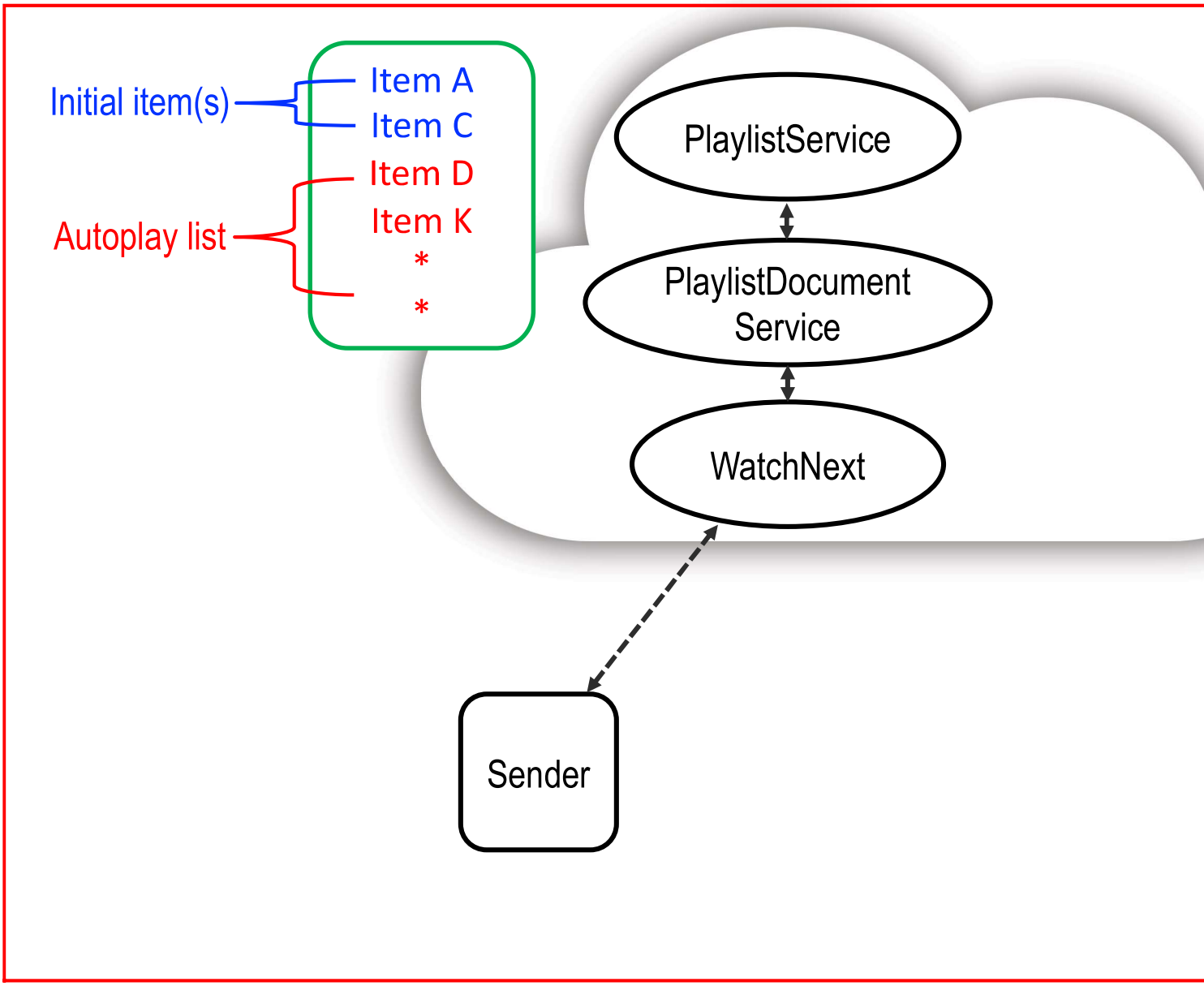
Cloud Services of YouTube Cloud Infrastructure



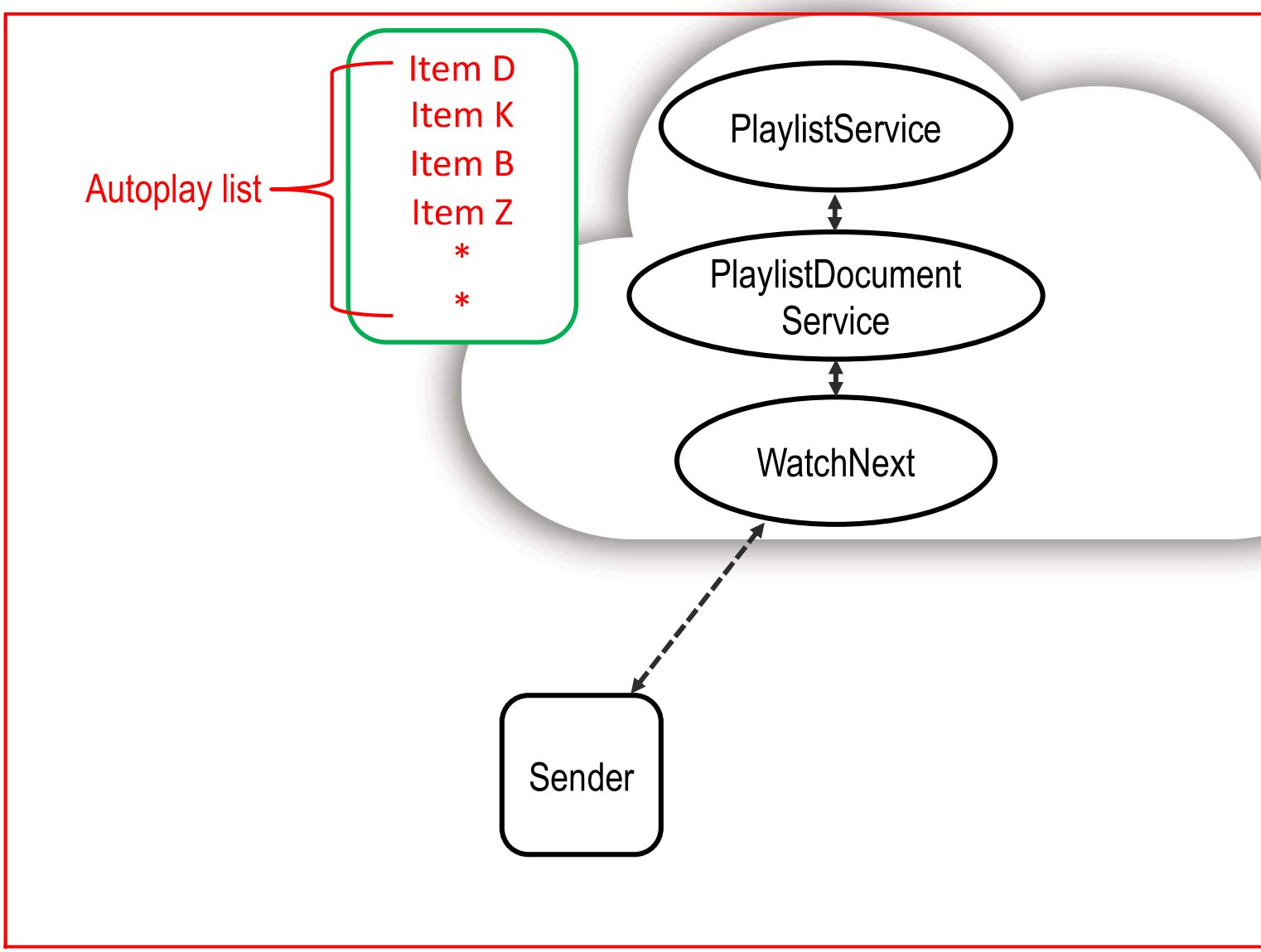
Sender Communicating with YouTube Cloud Infrastr



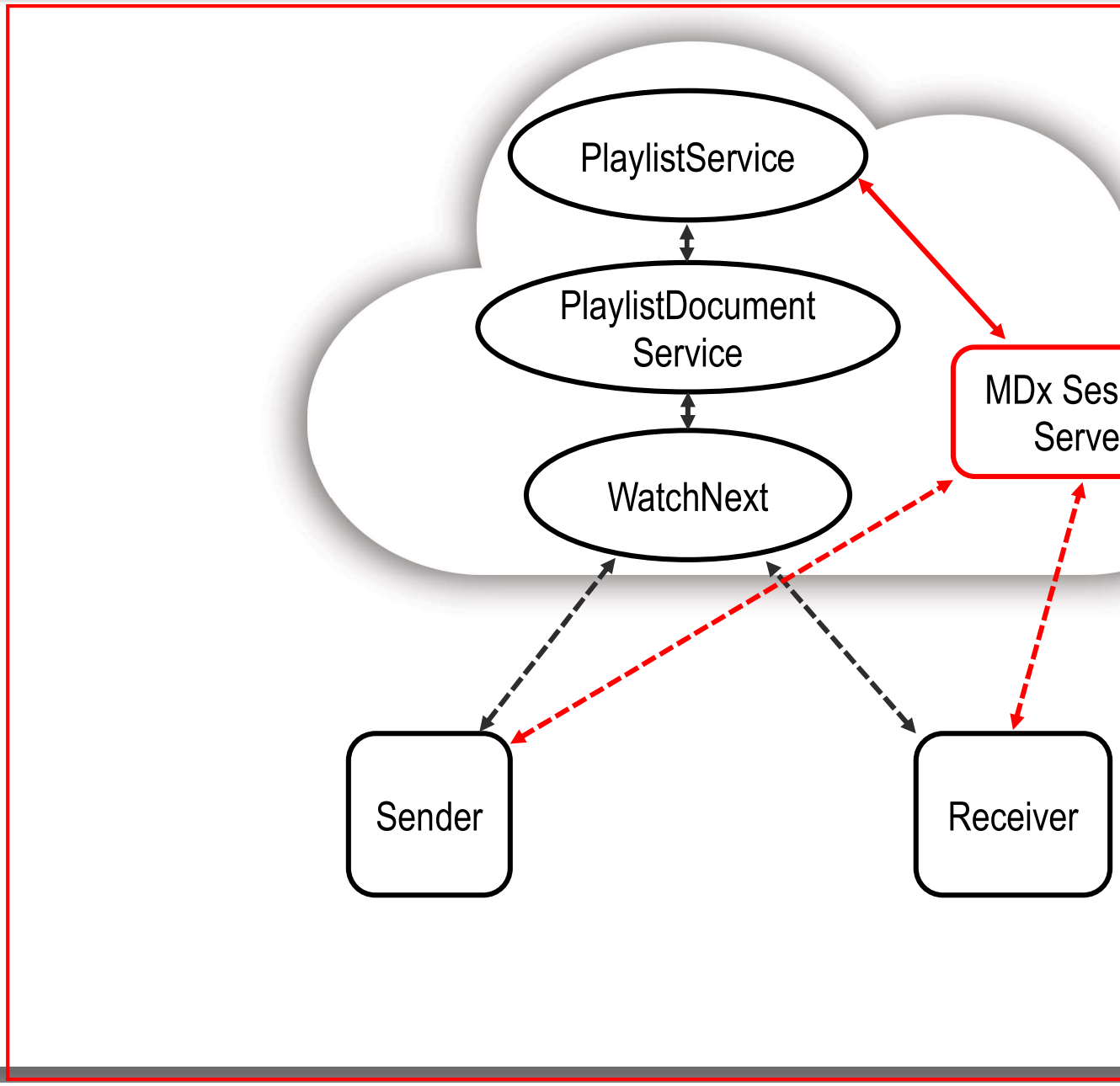
YouTube Cloud Infrastructure Providing Media Items



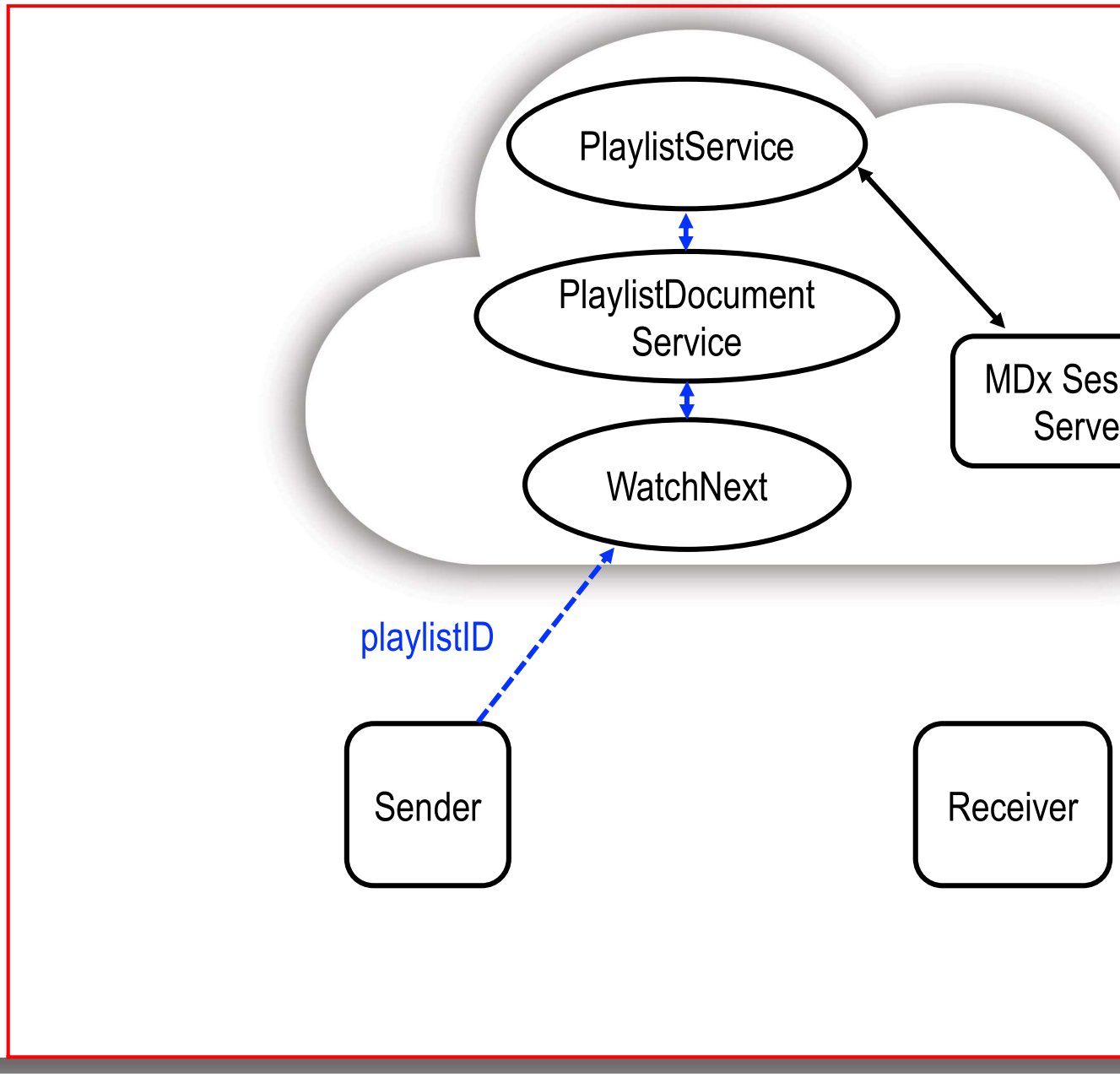
YouTube Cloud Infrastructure Providing Media Items



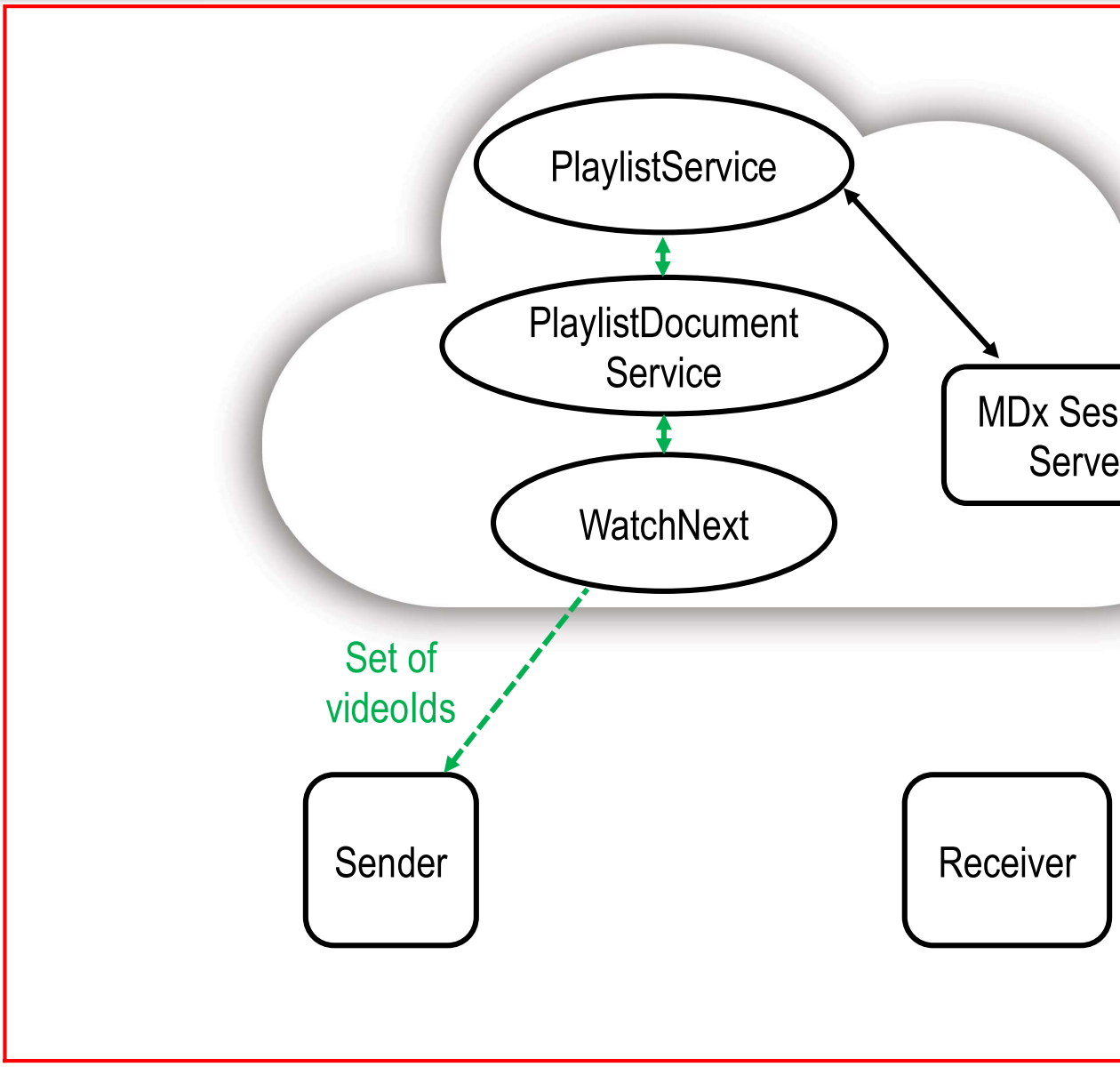
Casting and the YouTube Cloud Infrastructure



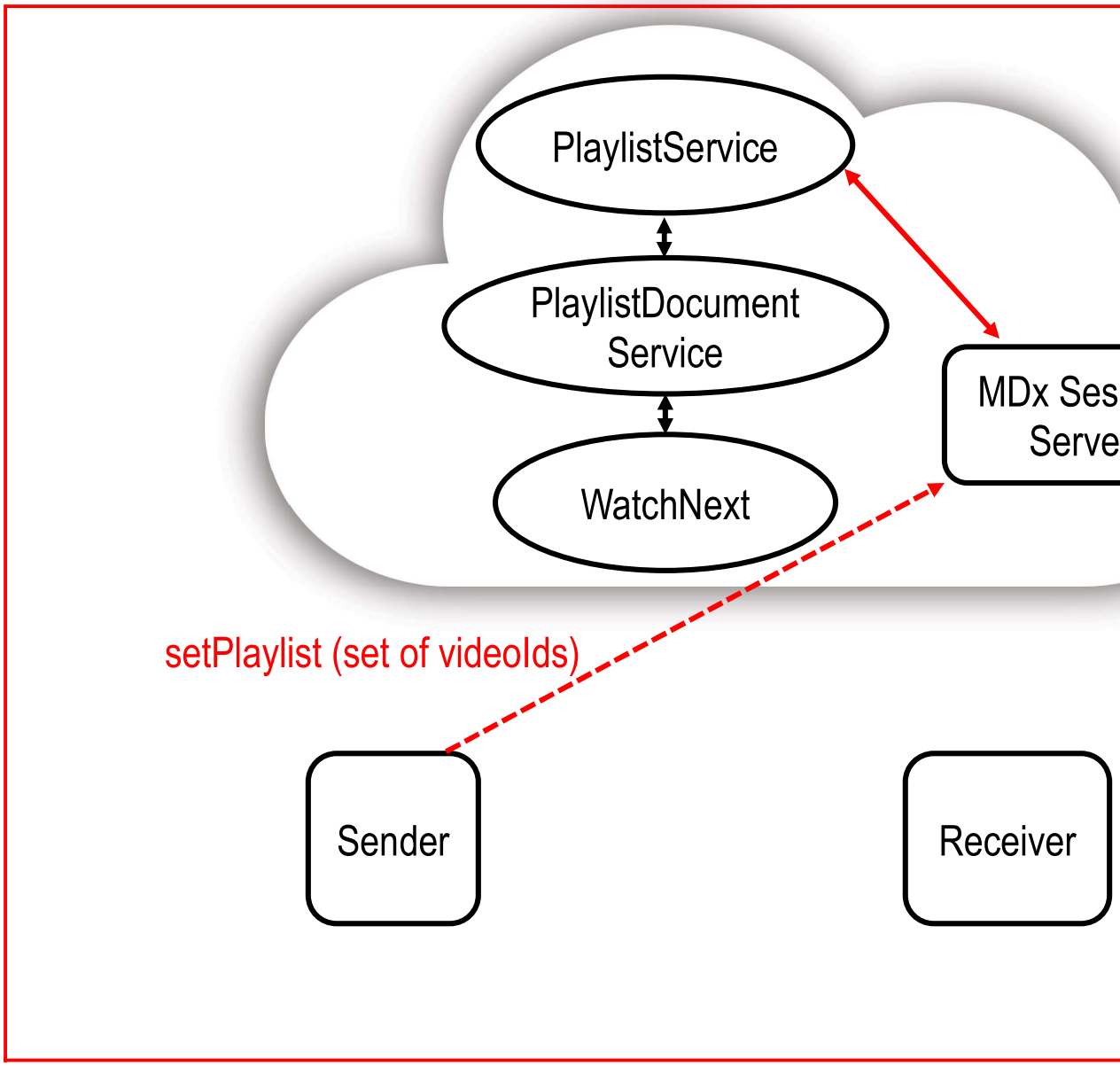
YouTube Music "Client-Side Expansion"



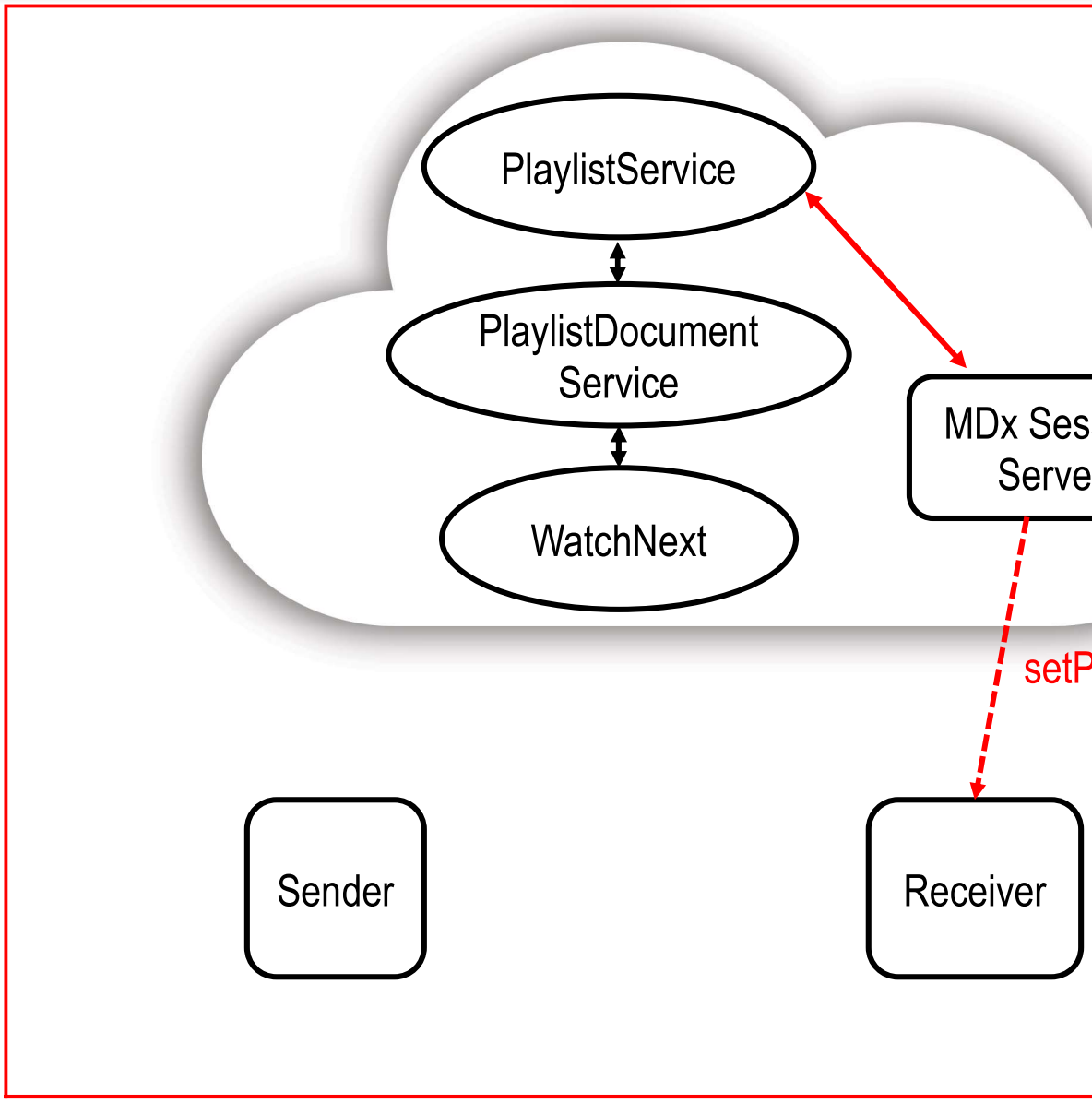
YouTube Music "Client-Side Expansion"



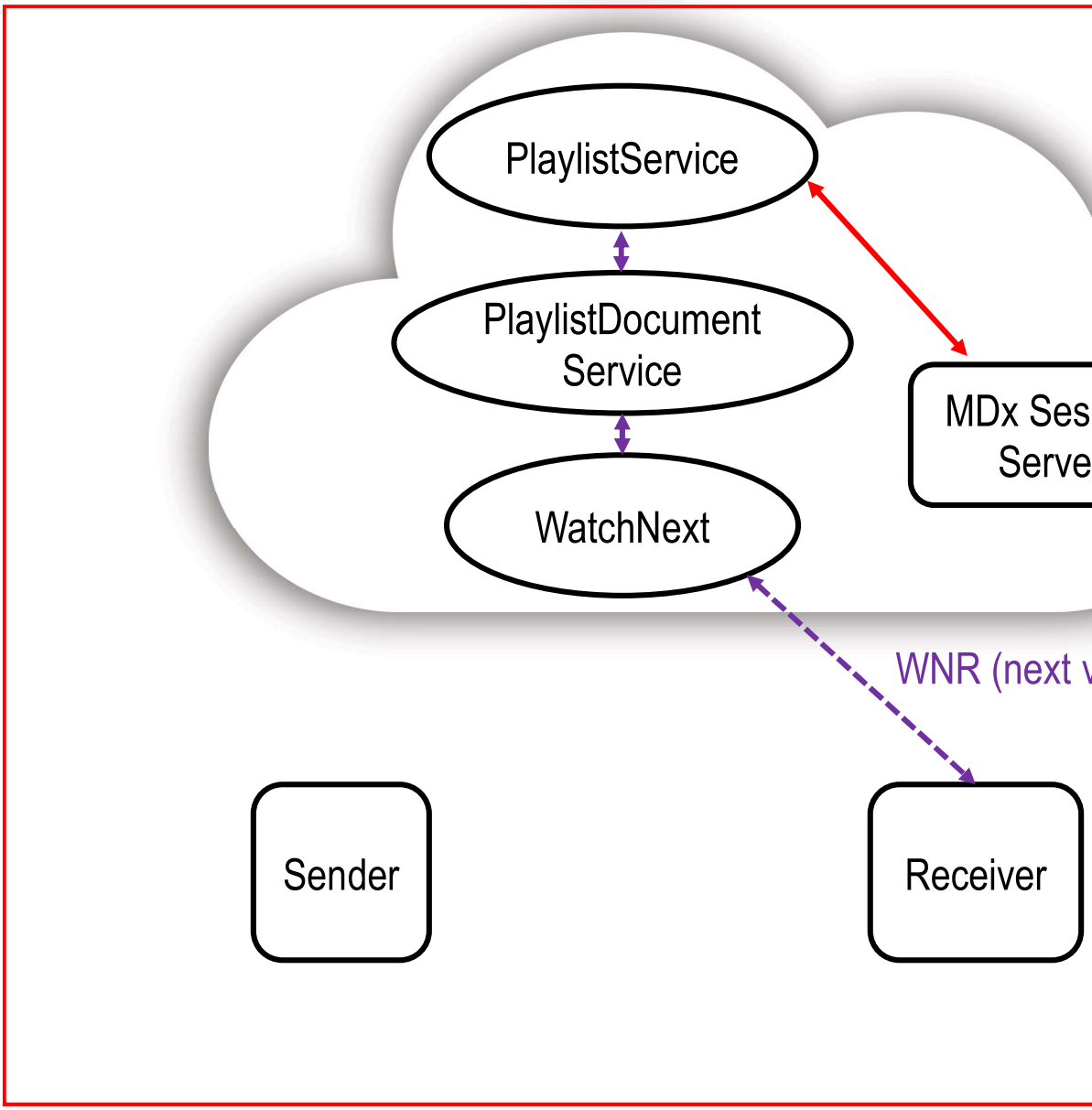
YouTube Music "Client-Side Expansion"



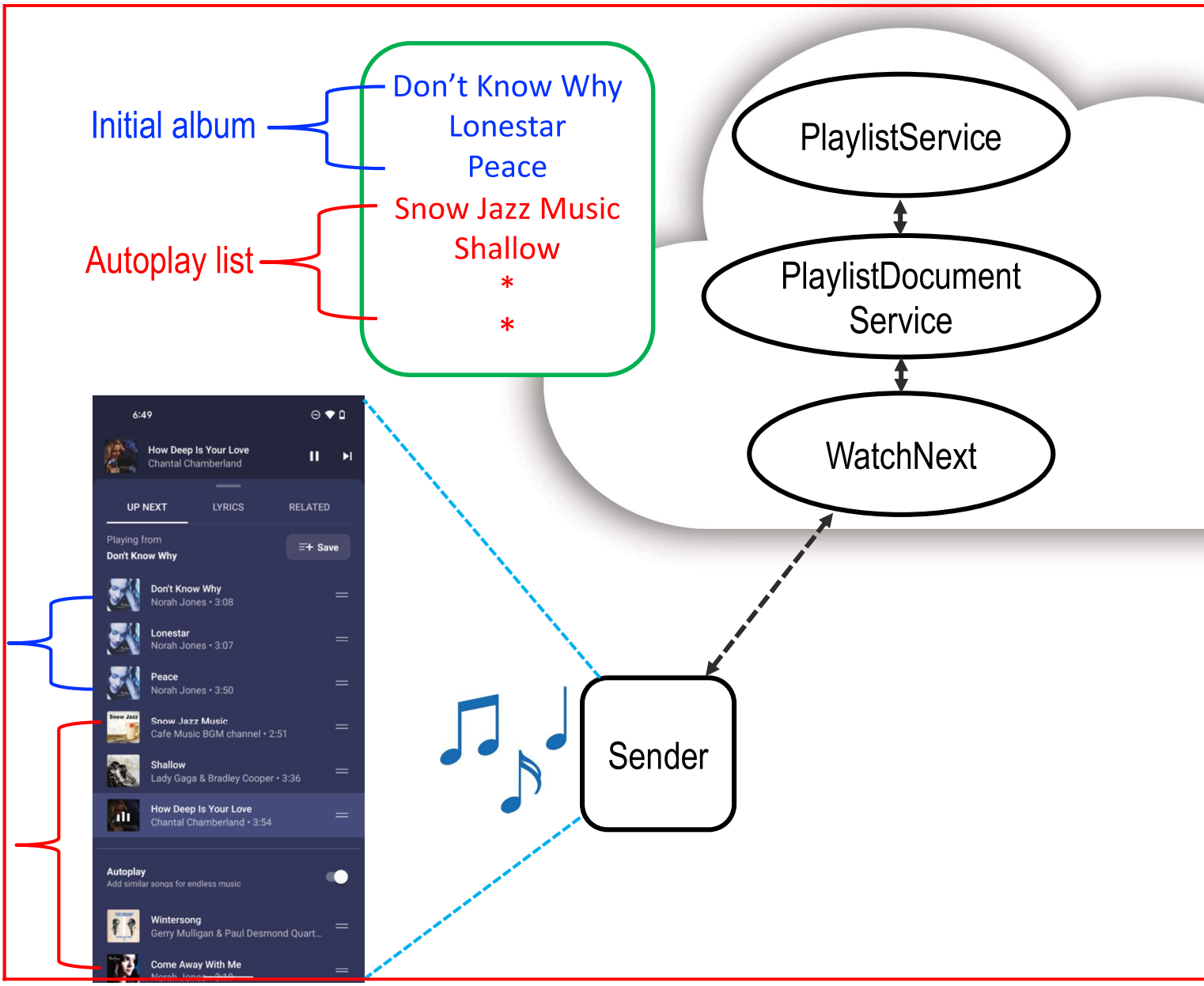
YouTube Music "Client-Side Expansion"



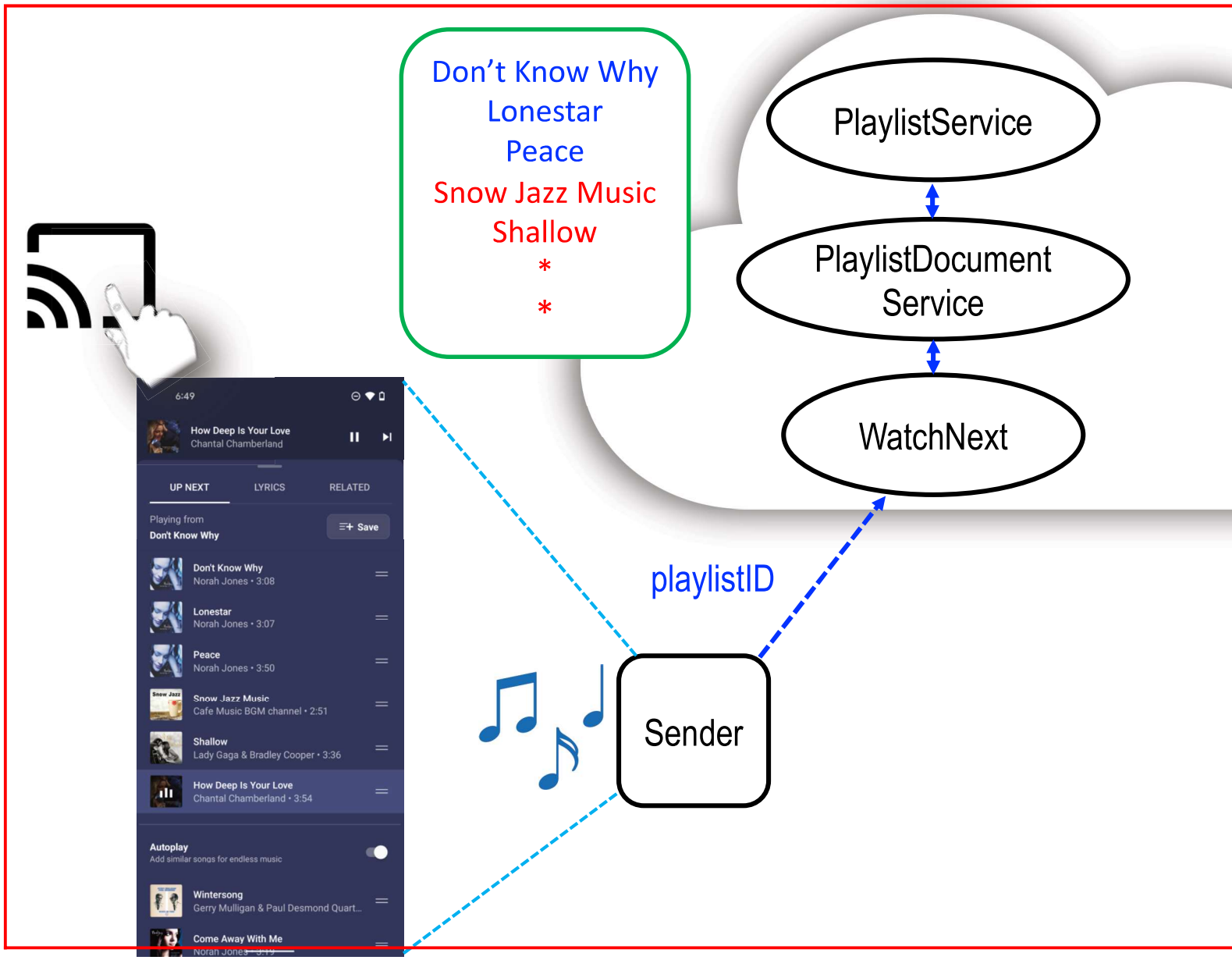
YouTube Music "Client-Side Expansion"



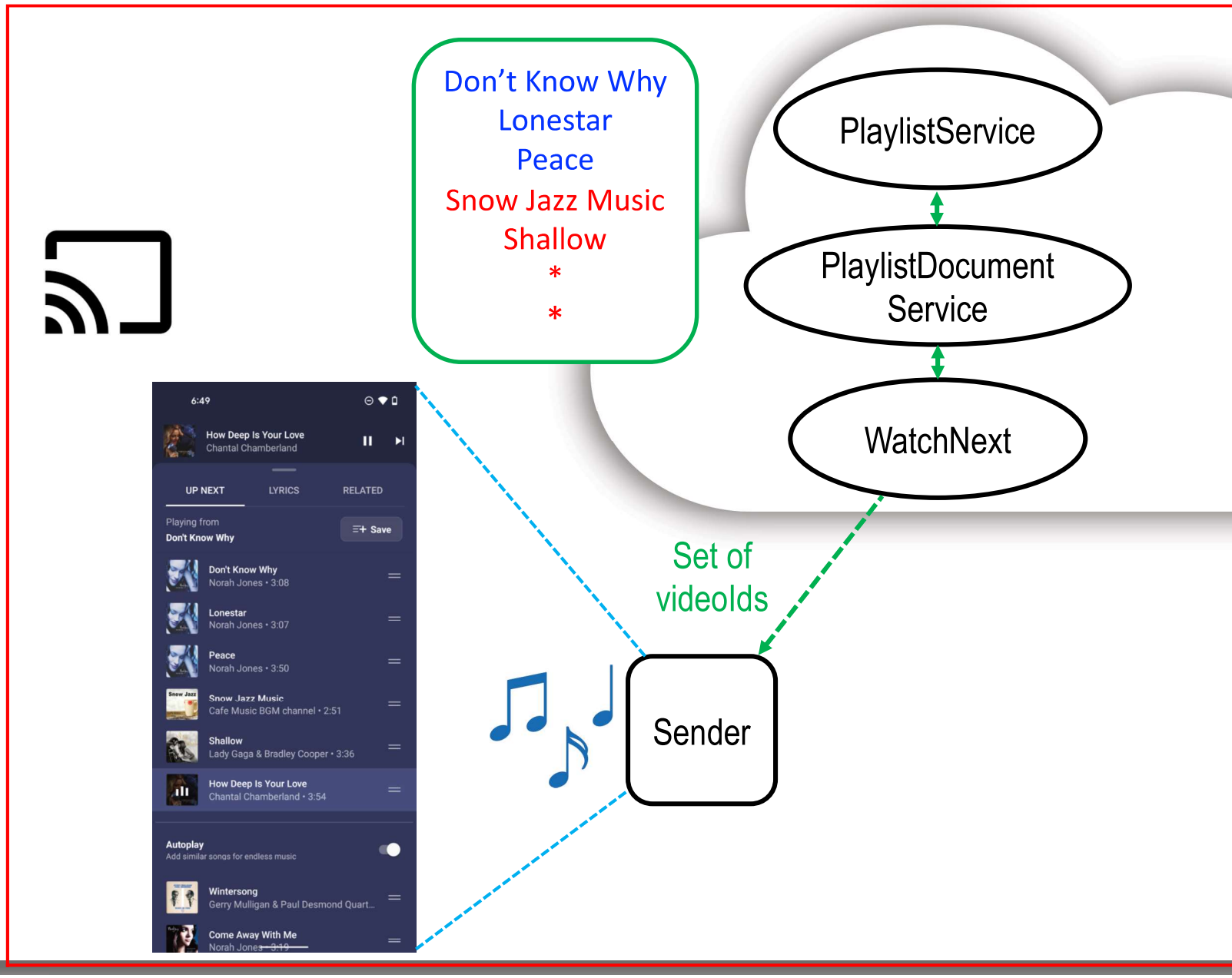
YouTube Music "Client-Side Expansion" Example



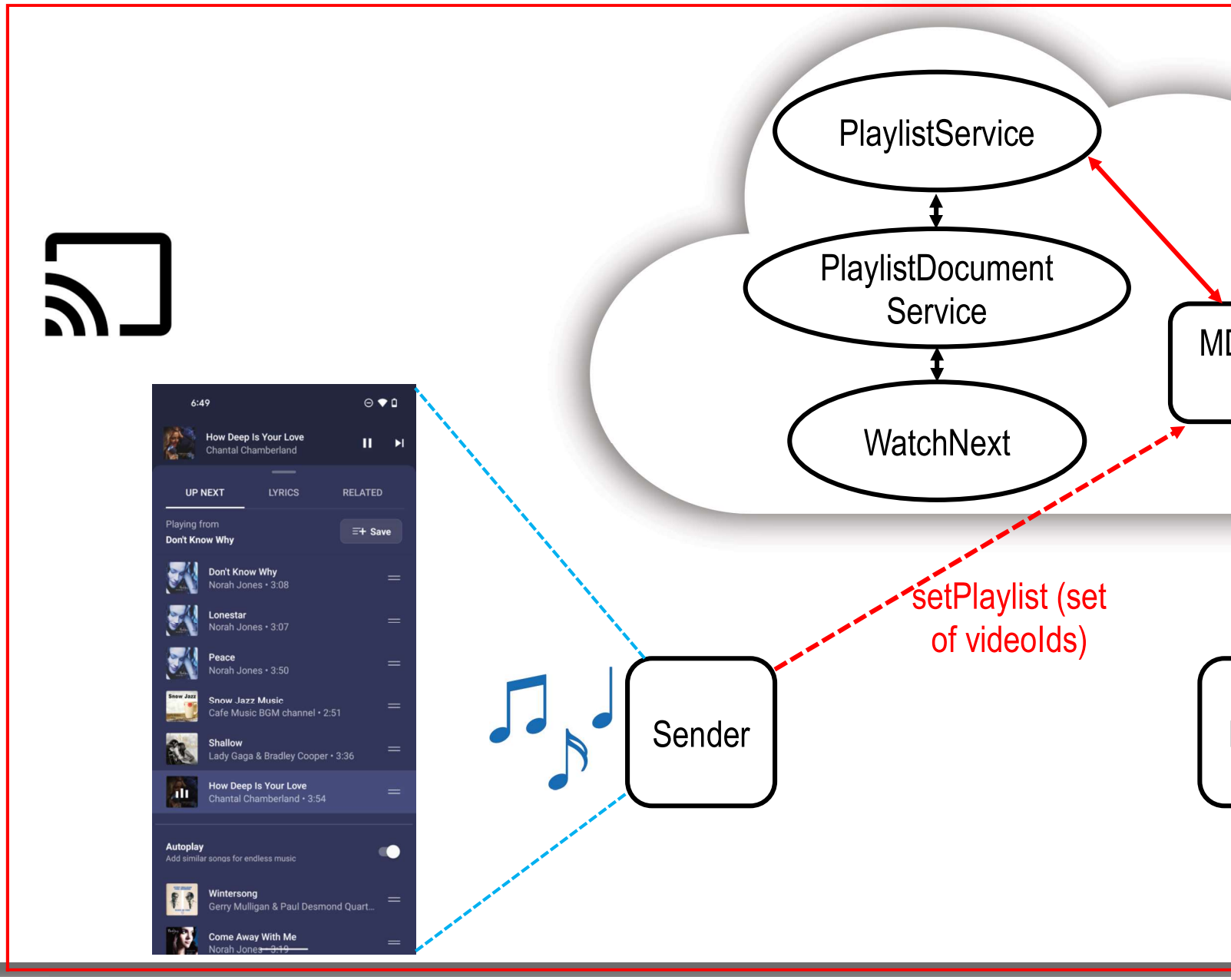
YouTube Music "Client-Side Expansion" Example



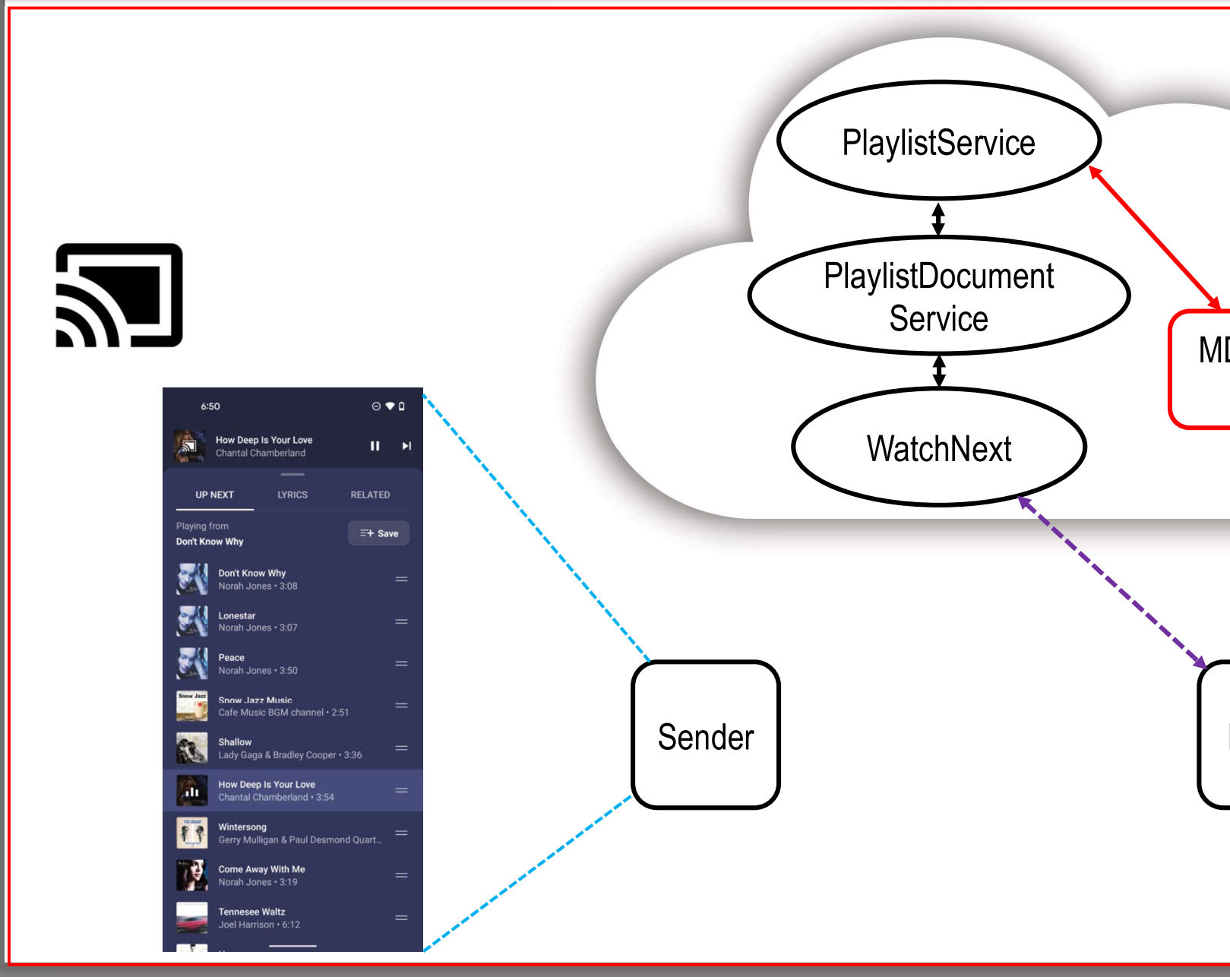
YouTube Music "Client-Side Expansion" Example



YouTube Music "Client-Side Expansion" Example



YouTube Music "Client-Side Expansion" Example



Cloud Provides List of Items for Playback When Non